



TUGAS AKHIR - KI141502

PENCARIAN RELASI OBJEK-OBJEK WARISAN BUDAYA INDONESIA BERDASARKAN KETERKAITAN INFORMASI TEMPORAL DENGAN MENGGUNAKAN ONTOLOGI

ALIEF YOGA PRIYANTO
NRP 5112100211

Dosen Pembimbing I
Sarwosri, S.Kom., M.T.

Dosen Pembimbing II
Nurul Fajrin Ariyani, S.Kom, M.Sc.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2016



TUGAS AKHIR - KI141502

PENCARIAN RELASI OBJEK-OBJEK WARISAN BUDAYA INDONESIA BERDASARKAN KETERKAITAN INFORMASI TEMPORAL DENGAN MENGGUNAKAN ONTOLOGI

**ALIEF YOGA PRIYANTO
NRP 5112100211**

**Dosen Pembimbing I
Sarwosri, S.Kom., M.T.**

**Dosen Pembimbing II
Nurul Fajrin Ariyani, S.Kom, M.Sc.**

**JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2016**

[Halaman ini sengaja dikosongkan]



UNDERGRADUATE THESES - KI141502

FINDING RELATIONS OF INDONESIAN CULTURAL HERITAGE OBJECTS BASED ON TEMPORAL INFORMATION LINKAGES WITH ONTOLOGY

**ALIEF YOGA PRIYANTO
NRP 5112100211**

**Supervisor I
Sarwosri, S.Kom., M.T.**

**Supervisor II
Nurul Fajrin Ariyani, S.Kom, M.Sc.**

**DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2016**

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

PENCARIAN RELASI OBJEK-OBJEK WARISAN BUDAYA INDONESIA BERDASARKAN KETERKAITAN INFORMASI TEMPORAL DENGAN MENGUNAKAN ONTOLOGI

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Manajemen Informasi
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh

ALIEF YOGA PRIYANTO

NRP : 5112 100 211

Disetujui oleh Dosen Pembimbing

1. Sarwosri, S.Kom., M.T

NIP. 19760809 200112 2 001

2. Nurul Fajrin Ariyani, S.Kom., M.Sc

NIP. 19860722 201504 2 003



SURABAYA

JULI, 2016

[Halaman ini sengaja dikosongkan]

Pencarian Relasi Objek-objek Warisan Budaya Indonesia Berdasarkan Keterkaitan Informasi Temporal dengan Menggunakan Ontologi

Nama Mahasiswa : ALIEF YOGA PRIYANTO
NRP : 5112100211
Jurusan : Teknik Informatika FTIF-ITS
Dosen Pembimbing 1 : Sarwosri, S.Kom., M.T.
Dosen Pembimbing 2 : Nurul Fajrin Ariyani, S.Kom, M.Sc.

Abstrak

Waktu adalah sebuah konsep penting dalam pencatatan objek-objek warisan budaya dan peristiwa sejarah. Keterangan waktu dapat melekat pada benda, tokoh maupun pada suatu kejadian sejarah. Contoh konsep waktu yang sering digunakan dalam pencatatan objek-objek warisan budaya dan sejarah adalah waktu interval (time-interval) dan waktu titik (time-point). Dalam pencatatan objek warisan budaya dan sejarah, informasi waktu disajikan dalam berbagai macam ragam penyimpanan. Informasi waktu disajikan dalam satuan waktu primitif seperti tanggal, bulan, tahun dan abad atau bisa juga disajikan dengan hanya menyebutkan keterangan waktu tertentu seperti zaman, era, masa serta keterangan waktu lainnya yang tidak diketahui secara pasti kapan terjadinya. Salah satu cara untuk mencari kedekatan waktu dengan satuan yang beragam adalah dengan memanfaatkan ontologi. Beberapa penelitian tentang ontologi untuk pencarian relasi waktu sudah ada seperti misalnya OWL-Time. Akan tetapi, skema ontologi OWL-Time masih terlalu luas dan tidak berfokus pada skema waktu warisan budaya sehingga dibutuhkan penelitian lebih lanjut untuk menangani hal ini.

Dalam Tugas Akhir ini dibuat sebuah skema ontologi untuk pencarian relasi antar entitas warisan budaya. Skema ini merupakan gabungan dari ontologi yang sudah ada yakni OWL-Time dan CIDOC-CRM. Penggabungan ontologi

tersebut dilakukan dengan Ontology Web Language (OWL) menggunakan tools Protégé. Berdasarkan uji coba yang dilakukan, skema ontologi ini dapat menghasilkan fakta-fakta baru mengenai kesamaan dan kedekatan waktu dari objek-objek yang diinputkan. Hal ini tentu saja dapat dijadikan suatu sumber untuk pembelajaran maupun penelitian berkaitan dengan warisan budaya. Hasil dari pencarian menggunakan skema ontologi ini ditampilkan dengan menggunakan aplikasi berbasis web. Dataset yang digunakan dalam proses uji coba terdiri dari 45 entitas temporal dan objek yang terkait yang merupakan hasil ekstraksi dari DBpedia ditambah 10 objek warisan budaya dari Monumen Nasional Indonesia.

Kata Kunci: Ontologi, OWL-Time, CIDOC-CRM, Cultural Heritage

Finding Relations of Indonesian Cultural Heritage Objects Based on Temporal Information Linkages With Ontology

Student's Name : ALIEF YOGA PRIYANTO
Student's ID : 5112100211
Department : Teknik Informatika FTIF-ITS
First Advisor : Sarwosri, S.Kom., M.T.
Second Advisor : Nurul Fajrin Ariyani, S.Kom, M.Sc.

Abstract

Time is an important concept in recording cultural heritage objects and historical events. Specification of time can be attached to objects, characters and on a historical event. Examples of the time concept which used in cultural heritage and historical events documentation is the time interval and the time point. In documenting cultural heritage and historical events, time is presented in variety diversion of storage. Information is presented in a primitive unit type such as date, month, year and century or it can be served with just mentioning a specific time information such as age, era, period and other time information that is never known exactly when its happened. One way to search the temporal information linkage with variety diversion of storage is using ontology. There are some research on searching time relation between cultural heritage objects with ontology such as OWL-Time. However OWL-Time is not focused on cultural heritage's time scheme so it needs more studies to handle this.

This final project created an ontology scheme for finding relationship between entities in cultural heritage. This scheme is a combination of existing ontologies namely OWL-Time and CIDOC-CRM. Merging ontologies are done with Web Ontology Language (OWL) using Protégé. Based on testing, this ontology this ontology scheme can generate new facts about the similarity and proximity of time of each object entered. It

can be used as a source for learning and research related to cultural heritage. The results of searches using an ontology schema is displayed using a web-based application. The dataset used in the testing process consists of 45 temporal entity and it's related objects that were extracted from DBpedia, also 10 cultural heritage objects from Monumen Nasional Indonesia.

Keywords: Ontology, OWL-Time, CIDOC-CRM, Cultural Heritage

KATA PENGANTAR

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

Puji syukur penulis kehadiran Tuhan YME karena berkat rahmat dan karunia-NYA penulis dapat menyelesaikan Tugas Akhir yang berjudul

PENCARIAN RELASI OBJEK-OBJEK WARISAN BUDAYA INDONESIA BERDASARKAN KETERKAITAN INFORMASI TEMPORAL DENGAN MENGUNAKAN ONTOLOGI

Tugas Akhir ini merupakan salah satu syarat untuk memperoleh gelar Sarjana Komputer di Jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember Surabaya.

Penulis ingin menyampaikan terima kasih yang sebesar-besarnya atas dukungan dan semangat yang diberikan dan membantu penulis baik secara langsung ataupun tidak dalam menyelesaikan Tugas Akhir ini. Penulis ingin mengucapkan terima kasih kepada

1. Allah SWT karena berkat rahmat dan karunia-Nya penulis berhasil menyelesaikan Tugas Akhir dengan baik.
2. Kedua orang tua, dan keluarga penulis, terima kasih atas doa dan bantuan moral dan material selama penulis belajar di Teknik Informatika ITS.
3. Ibu Sarwosri, S.Kom., M.T. selaku Dosen Pembimbing I dan Ibu Nurul Fajrin Ariyani, S.Kom, M.Sc. selaku pembimbing II Tugas Akhir yang telah memberikan bimbingan dan dukungan serta memberikan banyak waktu untuk berdiskusi dan memberi semangat dan motivasi kepada penulis untuk menyelesaikan Tugas Akhir.

4. Bapak Dr. Darlis Herumurti, S.Kom., M.Kom., selaku ketua jurusan Teknik Informatika ITS
5. Bapak Radityo Anggoro, S.Kom., M.Sc. selaku Koordinator Tugas Akhir di Teknik Informatika ITS.
6. Bapak dan Ibu Dosen di Jurusan Teknik Informatika yang telah memberikan ilmu selama penulis kuliah di Teknik Informatika
7. Seluruh Staf dan karyawan Teknik Informatika yang telah memberikan bantuan selama penulis kuliah di Teknik Informatika.
8. Rekan-rekan jurusan Teknik Informatika yang selalu memberi semangat dan motivasi kepada penulis hingga Tugas Akhir ini dapat terselesaikan.

Penulis Mohon maaf apabila terdapat kekurangan dalam penulisan Tugas Akhir ini. Kritik dan saran penulis harapkan untuk perbaikan dan pembelajaran di kemudian hari. Semoga Tugas Akhir ini dapat memberikan manfaat yang sebesar besarnya.

Surabaya, Juli 2016

Penulis

DAFTAR ISI

LEMBAR PENGESAHAN.....	v
<i>Abstrak</i>	vii
<i>Abstract</i>	ix
KATA PENGANTAR	xi
DAFTAR ISI	xiii
DAFTAR GAMBAR	xvii
DAFTAR TABEL	xxi
DAFTAR KODE SUMBER	xxiii
BAB I PENDAHULUAN	1
1.1 Latar Belakang.....	1
1.2 Rumusan Masalah	2
1.3 Batasan Masalah.....	3
1.4 Tujuan.....	3
1.5 Manfaat.....	3
1.6 Metodologi	4
BAB II TINJAUAN PUSTAKA.....	5
2.1 Warisan Budaya.....	5
2.2 Teori Temporal	6
2.3 Ontologi.....	6
2.4 RDF	9
2.5 OWL-Time	11
2.6 CIDOC-CRM	15
2.7 Semantik Web Rule Language (SWRL)	16
2.8 Google Refine.....	18
2.9 EasyRDF	19
2.10 SPARQL.....	19
BAB III METODOLOGI PEMECAHAN MASALAH	23
3.1 Analisis Data	23
3.1.1 Analisis Data DBpedia	23
3.1.2 Analisis Granularitas	24
3.2 Ekstraksi Data.....	25
3.2.1 Ekstraksi Data Menggunakan SPARQL Endpoint	25

3.2.2	Ekstraksi Data Manual.....	31
3.2.3	Transformasi Data Menggunakan Google Refine.....	35
3.3	Perancangan Ontologi.....	39
3.3.1	Skema Penyimpanan Informasi Temporal.....	40
3.3.2	Seleksi Kelas dan Properti	43
3.3.3	Penggabungan Ontologi	48
3.3.4	<i>Equivalent Class</i> dan <i>Equivalent Property</i>	49
3.3.5	Pengembangan Ontologi.....	51
3.4	Metode Pencarian Relasi	51
3.4.1	Pencarian Relasi <i>before</i>	54
3.4.2	Pencarian Relasi <i>after</i>	64
3.4.3	Pencarian Relasi <i>equals</i>	65
3.4.4	Pencarian Relasi <i>intervalMeets</i>	71
3.4.5	Pencarian Relasi <i>intervalMetBy</i>	72
3.4.6	Pencarian Relasi <i>intervalBefore</i>	72
3.4.7	Pencarian Relasi <i>intervalAfter</i>	72
3.4.8	Pencarian Relasi <i>intervalDuring</i>	73
3.4.9	Pencarian Relasi <i>intervalContains</i>	73
3.4.10	Pencarian Relasi <i>intervalStarts</i>	74
3.4.11	Pencarian Relasi <i>intervalStartedBy</i>	75
3.4.12	Pencarian Relasi <i>intervalFinishes</i>	75
3.4.13	Pencarian Relasi <i>intervalFinishedBy</i>	76
3.4.14	Pencarian Relasi <i>intervalOverlaps</i>	76
3.4.15	Pencarian Relasi <i>intervalOverlappedBy</i>	77
3.4.16	Pencarian Relasi <i>intervalEquals</i>	77
BAB IV ANALISIS DAN PERANCANGAN SISTEM.....		79
4.1	Analisis	79
4.1.1	Cakupan Permasalahan.....	79
4.1.2	Deskripsi Umum Sistem	79
4.1.3	Spesifikasi Kebutuhan Perangkat Lunak	80
4.1.4	Aktor.....	80
4.1.5	Kasus Penggunaan.....	81
4.2	Perancangan Antarmuka Pengguna	87
BAB V IMPLEMENTASI		91

5.1 Implementasi Fungsi	91
5.1.1 Fungsi <i>Combo Box</i>	91
5.1.2 Fungsi <i>Get Description</i>	91
5.1.3 Fungsi <i>Get Relations</i>	97
5.2 Implementasi Antarmuka	104
BAB VI PENGUJIAN DAN EVALUASI	107
6.1 Lingkungan Pengujian.....	107
6.2 Skenario Pengujian	107
6.2.1 Pengujian Ontologi.....	107
6.2.2 Pengujian Fungsionalitas.....	135
6.3 Evaluasi Pengujian	142
BAB VII KESIMPULAN DAN SARAN	145
7.1 Kesimpulan.....	145
7.2 Saran	145
DAFTAR PUSTAKA	147
BIODATA PENULIS	149

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

Gambar 2.1 Diagram Hirarki Ontologi CIDOC-CRM [8]	16
Gambar 3.1 Halaman <i>Resource</i> DBpedia	24
Gambar 3.2 Virtuoso SPARQL Query Editor DBpedia.....	25
Gambar 3.3 Penyimpanan Semantik Informasi Waktu pada DBpedia.....	30
Gambar 3.4 Rancangan <i>Skeleton Root</i> Entitas	37
Gambar 3.5 Rancangan <i>Skeleton Root Interval</i>	37
Gambar 3.6 Rancangan <i>Skeleton Root Instant</i> Awal	38
Gambar 3.7 Rancangan <i>Skeleton Root Instant</i> Akhir.....	38
Gambar 3.8 Penyimpanan Informasi Waktu pada DBpedia ..	42
Gambar 3.9 Skema Inti Ontologi Hasil Penggabungan	49
Gambar 3.10 Penambahan deskripsi “ <i>Equivalent To</i> ” pada Protégé.....	50
Gambar 3.11 Penambahan <i>Description</i> pada Protégé	53
Gambar 3.12 Penambahan <i>Characteristics</i> pada Protégé	53
Gambar 3.13 Ilustrasi Relasi <i>before</i> Kondisi 1	55
Gambar 3.14 Ilustrasi Relasi <i>before</i> Kondisi 2	55
Gambar 3.15 Ilustrasi Relasi <i>before</i> Kondisi 3	56
Gambar 3.16 Ilustrasi Relasi <i>before</i> Kondisi 4	57
Gambar 3.17 Ilustrasi Relasi <i>before</i> Kondisi 5	57
Gambar 3.18 Ilustrasi Relasi <i>before</i> Kondisi 6	58
Gambar 3.19 Ilustrasi Relasi <i>before</i> Kondisi 7	59
Gambar 3.20 Ilustrasi Relasi <i>before</i> Kondisi 8	60
Gambar 3.21 Ilustrasi Relasi <i>before</i> Kondisi 9	61
Gambar 3.22 Ilustrasi Relasi <i>before</i> Kondisi 10	62
Gambar 3.23 Ilustrasi Relasi <i>before</i> Kondisi 11	63
Gambar 3.24 Ilustrasi Relasi <i>before</i> Kondisi 12	64
Gambar 3.25 Penambahan Fungsi <i>Invers</i> pada Protégé	64
Gambar 3.26 Ilustrasi Relasi <i>equals</i> Kondisi 1	66
Gambar 3.27 Ilustrasi Relasi <i>equals</i> Kondisi 2	67
Gambar 3.28 Ilustrasi Relasi <i>equals</i> Kondisi 3	68
Gambar 3.29 Ilustrasi Relasi <i>equals</i> Kondisi 4	69
Gambar 3.30 Ilustrasi Relasi <i>equals</i> Kondisi 5	70

Gambar 3.31 Ilustrasi Relasi <i>equals</i> Kondisi 6	71
Gambar 3.32 Ilustrasi Relasi <i>intervalMeets</i>	71
Gambar 3.33 Ilustrasi Relasi <i>intervalMetBy</i>	72
Gambar 3.34 Ilustrasi Relasi <i>intervalBefore</i>	72
Gambar 3.35 Ilustrasi Relasi <i>intervalAfter</i>	73
Gambar 3.36 Ilustrasi Relasi <i>intervalDuring</i>	73
Gambar 3.37 Ilustrasi Relasi <i>intervalContains</i>	74
Gambar 3.38 Ilustrasi Relasi <i>intervalStarts</i>	74
Gambar 3.39 Ilustrasi Relasi <i>intervalStartedBy</i>	75
Gambar 3.40 Ilustrasi Relasi <i>intervalFinishes</i>	75
Gambar 3.41 Ilustrasi Relasi <i>intervalFinishedBy</i>	76
Gambar 3.42 Ilustrasi Relasi <i>intervalOverlaps</i>	76
Gambar 3.43 Ilustrasi Relasi <i>intervalOverlappedBy</i>	77
Gambar 3.44 Ilustrasi Relasi <i>intervalEquals</i>	78
Gambar 4.1 Diagram Kasus Penggunaan	81
Gambar 4.2 Diagram Aktivitas Kasus Penggunaan Memilih Entitas Melalui <i>Combo Box</i>	83
Gambar 4.3 Diagram Aktivitas Kasus Penggunaan Memilih Entitas Melalui Tautan	85
Gambar 4.4 Diagram Aktivitas Kasus Penggunaan Melihat Informasi Entitas	87
Gambar 4.5 Rancangan Antarmuka Halaman Utama 1	88
Gambar 4.6 Rancangan Antarmuka Halaman Utama 2.....	88
Gambar 5.1 Implementasi Antarmuka Halaman Utama 1....	105
Gambar 5.2 Implementasi Antarmuka Halaman Utama 2....	106
Gambar 6.1 Pengujian Kevalidan Relasi <i>before</i>	109
Gambar 6.2 Pengujian Kevalidan Relasi <i>after</i>	111
Gambar 6.3 Pengujian Kevalidan Relasi <i>equals</i>	112
Gambar 6.4 Pengujian Kevalidan Relasi <i>intervalMeets</i>	114
Gambar 6.5 Pengujian Kevalidan Relasi <i>intervalMetBy</i>	116
Gambar 6.6 Pengujian Kevalidan Relasi <i>intervalBefore</i>	117
Gambar 6.7 Pengujian Kevalidan Relasi <i>intervalAfter</i>	119
Gambar 6.8 Pengujian Kevalidan Relasi <i>intervalDuring</i>	121
Gambar 6.9 Pengujian Kevalidan Relasi <i>intervalContains</i> ..	122
Gambar 6.10 Pengujian Kevalidan Relasi <i>intervalStarts</i>	124

Gambar 6.11 Pengujian Kevalidan Relasi <i>intervalStartedBy</i>	126
Gambar 6.12 Pengujian Kevalidan Relasi <i>intervalFinishes</i>	127
Gambar 6.13 Pengujian Kevalidan Relasi <i>intervalFinishedBy</i>	129
Gambar 6.14 Pengujian Kevalidan Relasi <i>intervalOverlaps</i>	131
Gambar 6.15 Pengujian Kevalidan Relasi <i>intervalOverlappedBy</i>	133
Gambar 6.16 Pengujian Kevalidan Relasi <i>intervalEquals</i>	134
Gambar 6.17 Hasil Pengujian Fungsionalitas <i>Before</i>	137
Gambar 6.18 Hasil Pengujian Fungsionalitas <i>After</i> Bagian 1	138
Gambar 6.19 Hasil Pengujian Fungsionalitas <i>After</i> Bagian 2	139
Gambar 6.20 Hasil Pengujian Fungsionalitas <i>During</i>	140
Gambar 6.21 Hasil Pengujian Fungsionalitas <i>Interval</i>	142

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

Tabel 2.1 Komponen SWRL	17
Tabel 2.2 Komponen SWRL <i>Built-in</i>	18
Tabel 2.3 Contoh Hasil Keluaran <i>Query</i> SPARQL	21
Tabel 3.1 Hasil SPARQL Query Ekstraksi Data DBpedia	28
Tabel 3.2 Hasil Ekstraksi Data Tahap 1	31
Tabel 3.3 Hasil Ekstraksi Data Event dari Resource DBpedia Tahap 2	32
Tabel 3.4 Hasil Ekstraksi Data Anotasi dari ResourceDBpedia	34
Tabel 3.5 Contoh <i>Record</i> untuk Masukan Google Refine	35
Tabel 3.6 Daftar <i>Prefix</i> yang Digunakan untuk Perancangan <i>Skeleton</i>	36
Tabel 3.7 Penggunaan <i>Classes</i> dari OWL-Time dan CIDOC-CRM	44
Tabel 3.8 Penggunaan <i>Object Property</i> dari OWL-Time dan CIDOC-CRM	44
Tabel 3.9 Penggunaan <i>Data Property</i> dari OWL-Time dan CIDOC-CRM	48
Tabel 3.10 Daftar <i>Equivalent Classes</i>	49
Tabel 3.11 Daftar <i>Equivalent Properties</i>	49
Tabel 3.12 Penambahan Kelas	51
Tabel 3.13 Relasi Properti dengan Menggunakan <i>Description</i>	52
Tabel 3.14 Relasi Properti dengan Menggunakan <i>Characteristics</i>	53
Tabel 4.1 Kebutuhan Fungsional Sistem	80
Tabel 4.2 Daftar Kode Diagram Kasus Penggunaan	81
Tabel 4.3 Spesifikasi Kasus Penggunaan Memilih Entitas Melalui <i>Combo Box</i>	82
Tabel 4.4 Spesifikasi Kasus Penggunaan Memilih Entitas Melalui Tautan	84

Tabel 4.5 Spesifikasi Kasus Penggunaan Melihat Informasi Entitas	85
Tabel 4.6 Penjelasan Atribut Perancangan Antarmuka Halaman Utama	89
Tabel 6.1 Pengujian Kevalidan Relasi <i>before</i>	108
Tabel 6.2 Pengujian Kevalidan Relasi <i>after</i>	109
Tabel 6.3 Pengujian Kevalidan Relasi <i>equals</i>	111
Tabel 6.4 Pengujian Kevalidan Relasi <i>intervalMeets</i>	113
Tabel 6.5 Pengujian Kevalidan Relasi <i>intervalMetBy</i>	114
Tabel 6.6 Pengujian Kevalidan Relasi <i>intervalBefore</i>	116
Tabel 6.7 Pengujian Kevalidan Relasi <i>intervalAfter</i>	118
Tabel 6.8 Pengujian Kevalidan Relasi <i>intervalDuring</i>	119
Tabel 6.9 Pengujian Kevalidan Relasi <i>intervalContains</i>	121
Tabel 6.10 Pengujian Kevalidan Relasi <i>intervalStarts</i>	123
Tabel 6.11 Pengujian Kevalidan Relasi <i>intervalStartedBy</i>	124
Tabel 6.12 Pengujian Kevalidan Relasi <i>intervalFinishes</i>	126
Tabel 6.13 Pengujian Kevalidan Relasi <i>intervalFinishedBy</i>	128
Tabel 6.14 Pengujian Kevalidan Relasi <i>intervalOverlaps</i>	130
Tabel 6.15 Pengujian Kevalidan Relasi <i>intervalOverlappedBy</i>	131
Tabel 6.16 Pengujian Kevalidan Relasi <i>intervalEquals</i>	133
Tabel 6.17 Pengujian Fungsionalitas <i>Before</i>	135
Tabel 6.18 Pengujian Fungsionalitas <i>After</i>	137
Tabel 6.19 Pengujian Fungsionalitas <i>During</i>	139
Tabel 6.20 Pengujian Fungsionalitas <i>Interval</i>	141
Tabel 6.21 Evaluasi Pengujian Ontologi	143
Tabel 6.22 Evaluasi Pengujian Fungsionalitas	144

DAFTAR KODE SUMBER

Kode Sumber 2.1 Topologi Kelas <i>TemporalEntity</i> dalam OWL	11
Kode Sumber 2.2 Topologi Kelas <i>DurationDescription</i> dalam OWL.....	13
Kode Sumber 2.3 Deskripsi <i>Datetime</i> dengan Properti <i>inXSDDatetime</i> dan <i>inDatetime</i>	15
Kode Sumber 2.4 Contoh <i>Query</i> SPARQL.....	20
Kode Sumber 3.1 SPARQL Query untuk Ekstraksi Data DBpedia.....	27
Kode Sumber 3.2 Contoh Hasil Transformasi Tabel Ke Dalam Format RDF/XML	39
Kode Sumber 5.1 Fungsi <i>Combo Box</i>	91
Kode Sumber 5.2 <i>Get Related Entity</i>	92
Kode Sumber 5.3 <i>Get</i> Granularitas Waktu Mulai	95
Kode Sumber 5.4 <i>Get</i> Granularitas Waktu Selesai	97
Kode Sumber 5.5 <i>Get intervalEquals</i>	97
Kode Sumber 5.6 <i>Get intervalBefore</i>	98
Kode Sumber 5.7 <i>Get intervalAfter</i>	98
Kode Sumber 5.8 <i>Get intervalOverlaps</i>	99
Kode Sumber 5.9 <i>Get intervalOverlappedBy</i>	99
Kode Sumber 5.10 <i>Get intervalContains</i>	100
Kode Sumber 5.11 <i>Get intervalDuring</i>	101
Kode Sumber 5.12 <i>Get intervalMeets</i>	101
Kode Sumber 5.13 <i>Get intervalMetBy</i>	102
Kode Sumber 5.14 <i>Get intervalStarts</i>	102
Kode Sumber 5.15 <i>Get intervalStartedBy</i>	103
Kode Sumber 5.16 <i>Get intervalFinishes</i>	103
Kode Sumber 5.17 <i>Get intervalFinishedBy</i>	104

[Halaman ini sengaja dikosongkan]

BAB I

PENDAHULUAN

Pada bab ini dibahas mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi, dan sistematika laporan tugas akhir. Diharapkan dari penjelasan dalam bab ini gambaran Tugas Akhir secara umum dapat dipahami.

1.1 Latar Belakang

Warisan budaya adalah peninggalan yang berupa benda maupun non-benda yang merupakan jati diri suatu masyarakat yang diwariskan dari generasi sebelumnya dan yang dilestarikan untuk generasi yang akan datang. Warisan budaya dapat berupa benda material seperti candi, monumen, artefak, fosil, dan prasasti. Warisan budaya tidak hanya terbatas pada benda-benda material yang dapat dilihat dan disentuh. Warisan budaya juga terdiri dari unsur-unsur non-material seperti tradisi, sejarah tak tertulis, seni pertunjukan, ritual, pengetahuan dan keterampilan yang ditransmisikan dari generasi ke generasi dalam suatu komunitas.

Seiring berkembangnya zaman, pola hidup masyarakat mulai berkembang ke arah yang lebih modern dan mulai melepaskan dari akarnya yaitu budaya asli mereka sendiri. Hal tersebut terjadi disebabkan oleh banyak faktor salah satunya adalah perkembangan teknologi informasi yang sangat pesat. Teknologi informasi seolah-olah telah menjadi pedoman hidup bagi masyarakat sekarang ini. Namun demikian, di sisi lain teknologi informasi juga menjadi kekuatan besar yang dapat dimanfaatkan untuk menjaga kelestarian warisan budaya. Pendokumentasian data warisan budaya dengan memanfaatkan teknologi merupakan salah satu langkah dalam usaha pelestarian warisan budaya.

Waktu adalah sebuah konsep penting dalam pencatatan warisan budaya. Dengan konsep waktu, budayawan maupun

peneliti sejarah dapat lebih mudah mempelajari keruntutan warisan budaya dalam garis waktu. Oleh karena itu penting untuk mencocokkan query dan anotasi temporal dengan memeriksa kesamaan atau kedekatannya. Contoh konsep waktu yang digunakan untuk anotasi dari benda-benda peninggalan budaya adalah waktu interval (*time-interval*) dan waktu titik (*time-point*). Salah satu masalah adalah banyaknya kasus yang menggunakan metode skema penyimpanan waktu yang kurang tepat. Sebagai contoh “Abad 20M” dan “Zaman Kolonial” disimpan sebagai *string* maupun sebagai *integer* sehingga tidak dapat diketahui bahwa sebenarnya kedua keterangan temporal tersebut memiliki kedekatan dan keterkaitan. Dibutuhkan skema pencatatan waktu untuk mempermudah meruntutkan peristiwa-peristiwa sejarah, penemuan, atau apapun yang berkaitan dengan peninggalan warisan budaya. Ontologi dapat diterapkan untuk dijadikan solusi atas permasalahan pencatatan semantik informasi waktu pada dokumentasi warisan budaya.

Pada tugas akhir ini akan dibuat skema ontologi untuk membantu dalam dokumentasi peninggalan warisan budaya. Informasi objek-objek warisan budaya akan ditampilkan menurut pencarian berdasarkan keterkaitan waktu.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam tugas akhir dapat dipaparkan sebagai berikut.

1. Bagaimana skema Time Ontology untuk penyimpanan semantik dari informasi waktu yang diterapkan pada studi kasus warisan budaya?
2. Bagaimana standarisasi penyimpanan data waktu agar dapat menampilkan data yang benar ketika dilakukan *reasoning*?
3. Bagaimana cara menampilkan informasi warisan-warisan budaya yang saling terkait hubungan waktunya?

1.3 Batasan Masalah

Batasan masalah yang terdapat pada Tugas Akhir ini, yaitu sebagai berikut:

1. Skema ontologi waktu akan dibangun menggunakan bahasa RDF dan OWL.
2. Pembangunan ontologi dilakukan dengan menggunakan aplikasi Protégé.
3. Dataset yang digunakan adalah peninggalan warisan budaya dari artefak dan peristiwa sejarah di Indonesia.
4. Data yang digunakan berasal dari di DBpedia.
5. Antarmuka yang disediakan hanya berfungsi untuk menampilkan *inference* data hasil *reasoning*.

1.4 Tujuan

Tujuan dari pembuatan tugas akhir ini, antara lain:

1. Membuat model penyimpanan semantik informasi waktu yang melekat pada warisan budaya menggunakan konsep ontologi
2. Mencari hubungan antar objek pada warisan budaya yang memiliki keterkaitan berdasarkan informasi temporalnya secara otomatis menggunakan fungsi *rule* SWRL (*Semantic Web Rule Language*).
3. Menggunakan teknologi Google Refine untuk memetakan data ke dalam skema ontologi.
4. Membuat program berbasis web untuk menampilkan *instances*.

1.5 Manfaat

Tugas akhir ini diharapkan dapat menyeragamkan penyimpanan informasi waktu dan mempermudah pengaksesan data peninggalan budaya berdasarkan hubungan waktunya.

1.6 Metodologi

Metodologi yang dipakai pada pengerjaan Tugas Akhir ini adalah sebagai berikut:

1. Penyusunan Proposal Tugas Akhir
Tahap awal yang dilakukan dalam pengerjaan Tugas Akhir ini adalah penyusunan proposal Tugas Akhir. Di dalam proposal diajukan suatu gagasan pembuatan skema ontologi untuk mencari relasi antar objek warisan budaya.
2. Studi Literatur
Pada tahap ini dilakukan pencarian, pengumpulan, penyaringan, pemahaman, dan pembelajaran literatur yang berhubungan dengan skema ontologi OWL-Time, CIDOC-CRM, SWRL, SPARQL dan *library* EasyRDF. Literatur yang digunakan meliputi: jurnal, dan dokumentasi internet.
3. Implementasi dan pembuatan perangkat lunak
Pada tahap ini dilakukan implelementasi perangkat lunak sesuai dengan rancangan perangkat lunak yang dibuat yang terdiri dari tahap pembentukan skema ontologi, *input* data, perancangan *rule* dan pembuatan aplikasi penampil data.
4. Uji coba dan Evaluasi
Pada tahap ini dilakukan uji coba terhadap perangkat lunak yang telah dibuat untuk mengetahui kemampuan skema model ontologi mengolah data berdasarkan *rule* yang telah dibuat.
5. Penyusunan Laporan Tugas Akhir
Pada tahap ini dilakukan penyusunan laporan pengerjaan Tugas Akhir yang berisi dasar teori, dokumentasi dari perangkat lunak, dan hasil yang diperoleh.

BAB II

TINJAUAN PUSTAKA

Pada bab ini dibahas mengenai teori-teori yang menjadi dasar dari pembuatan Tugas Akhir.

2.1 Warisan Budaya

Warisan budaya adalah hasil budaya fisik (*tangible*) dan non fisik (*intangible*) dari peranan masa lalu [1]. Warisan budaya merupakan bagian dari Pusaka Budaya suatu bangsa. Pusaka Budaya merupakan karya cipta, rasa dan karsa yang istimewa dari beragam suku bangsa di Nusantara baik yang dimiliki secara masing-masing ataupun sebagai kesatuan bangsa Indonesia. Pusaka Budaya juga dapat terbentuk dari interaksi antar suku bangsa sepanjang sejarah keberadaannya.

Menurut J.J. Hoenigman, kebudayaan dibedakan menjadi tiga menurut wujudnya, yaitu:

1. Gagasan (Wujud Ideal)

Gagasan adalah kebudayaan yang berwujud ide-ide, gagasan, nilai, norma, peraturan dan sebagainya yang sifatnya abstrak dan tidak dapat diraba maupun disentuh. Wujud kebudayaan ini tersirat, terletak di dalam pemikiran warga masyarakat. Kebudayaan gagasan bisa terdapat dalam buku atau karangan apabila masyarakat menyatakan gagasan mereka itu dalam bentuk tulisan.

2. Aktivitas

Aktivitas adalah tindakan berpola dari manusia dalam masyarakat. Wujud kebudayaan aktivitas sering disebut sebagai sistem sosial. Sistem sosial terdiri dari aktivitas-aktivitas manusia yang saling berinteraksi, mengadakan kontak, serta bergaul dengan manusia lain dengan pola-pola tertentu berdasarkan norma adat kelakuan. Sistem sosial memiliki sifat yang konkret karena terjadi di kehidupan sehari-hari dan dapat diamati dan didokumentasikan.

3. Artefak

Artefak adalah wujud fisik kebudayaan yang merupakan hasil dari aktivitas, perbuatan, dan karya manusia dalam masyarakat. Artefak berupa benda atau hal-hal nyata yang dapat diraba, diamati, dan didokumentasikan. Wujud kebudayaan ini sifatnya paling konkret di antara ketiga wujud¹.

2.2 Teori Temporal

Dalam bahasa Inggris waktu (*time*) memiliki banyak arti. Waktu dapat diartikan sebagai sebuah konsep yang sering digunakan dalam pernyataan linguistik dan juga merupakan topik sentral dalam filosofi yang memiliki domain-domain berbeda untuk penyampaian fisik menggunakan representasi pengetahuan dan pemikiran [2]. Hayes mengidentifikasi enam konsep waktu yang saling terkait satu sama lain. Dua diantaranya paling mendukung representasi waktu warisan budaya, antara lain:

1. Interval waktu (*time-interval*): bagian waktu yang terletak pada rangkaian kesatuan *temporal* (atau plenum waktu) yang digunakan sebagai dasar untuk teori waktu.
2. Titik waktu (*time-point*): melambangkan suatu “titik” pada waktu yang mendukung teori *temporal*. Konsep ini terkadang dipahami sebagai titik koordinat waktu yang tidak memiliki durasi dan berguna dalam plenum waktu.

2.3 Ontologi

Sebuah ontology merupakan definisi dari pengertian dasar dan relasi perbendaharaan kata dari sebuah area yang memiliki aturan dari kombinasi istilah dan relasi untuk mendefinisikan perbendaharaan kata [3]. Menurut Gruber [4], ontologi adalah suatu spesifikasi baku dan jelas dari sebuah konseptualisasi. Ontology merupakan suatu teori tentang makna dari suatu objek, properti dari suatu objek, serta relasi

¹ <https://id.wikipedia.org/wiki/Budaya>

objek tersebut yang mungkin terjadi pada suatu domain pengetahuan. Ontologi menjelaskan tentang konsep dalam *domain (class)* tertentu, *properties* pada masing-masing konsep, atribut dalam sebuah konsep, dan batasan-batasan konsep (*rule*). Sebuah ontologi dengan *instance* dapat membentuk sebuah dasar pengetahuan.

1. Komponen Ontologi

Banyaknya variasi struktur ontologi tergantung dari penggunaan bahasa ontologi, termasuk sintaksis yang digunakan. Pada dasarnya, tidak ada fungsionalitas apapun yang dilakukan oleh ontologi, ontologi yang menggunakan fungsi perhitungan dan lainnya tergantung dari data yang ada di dalam ontologi tersebut dan aplikasi yang digunakan. Ontologi memiliki beberapa komponen, diantaranya:

- a) Konsep (*Concept*), merupakan deskripsi dari tugas, fungsi, aksi, strategi, dan sebagainya. Dalam *file* berbasis ontologi, *concept* dikenal sebagai *class*. *Class* dalam ontologi mencakup *subclass* dan *superclass*. *Subclass* mewarisi atribut dari *superclass*-nya sedangkan *superclass* mewariskan atribut kepada *subclass*-nya.
- b) Relasi (*Relation*), merupakan representasi sebuah interaksi antara konsep dari sebuah *domain*. Misalnya relasi *binary*, yaitu *subclass-of* dan *same-as*.
- c) Fungsi (*Function*), yaitu relasi khusus dimana relasi dari suatu elemen adalah unik untuk elemen lainnya.
- d) Aksiom (*Axioms*), berfungsi untuk memodelkan pernyataan yang selalu benar.
- e) *Instance*, digunakan untuk merepresentasikan elemen atau objek. *Instance* merupakan member dari suatu *class*.

2. Bahasa Ontologi

Ontologi dapat digunakan dengan mengekspresikan notasi yang nyata. Bahasa ontologi dapat dikatakan sebagai

bahasa formal dari sebuah pengembangan ontologi, Komponen yang menjadi struktur ontologi, antara lain:

- XML (*Extensible Markup Language*), mendukung *output* dokumen dalam bentuk terstruktur, kecuali XML yang menggunakan *semantic constraints*. Untuk pembatasan struktur dokumen, bahasa yang digunakan adalah XML *Schema*.
- RDF (*Resource Description Framework*), merupakan bahasa yang dapat memodelkan data untuk objek yang memiliki hubungan dengan objek lainnya. *Semantic* yang sederhana juga disediakan oleh bahasa ini untuk model data tersebut. Selain dapat menyimpan berkas dengan format RDF, bahasa RDF juga mendukung penyajian data dalam struktur XML. RDF memiliki skema yang selanjutnya disebut sebagai RDF *Schema*. *Properties* dan *classes* dari sumber RDF merupakan kosakata yang dijelaskan dalam skema tersebut. Skema tersebut dilengkapi dengan sebuah hirarki *semantics* yang menyamaratakan *properties* dan *classes*.
- OWL (*Ontology Web Language*), melengkapi kosakata yang dimiliki RDF. Selain menjelaskan *properties* dan *classes*, OWL mendeskripsikan relasi antara *classes* (misalkan *disjointness*), kardinalitas (misalkan '?tepat satu'), dan *equality*. OWL juga menyediakan berbagai tipe dan karakteristik (misalkan *symmetry*) dari *properties*.

Proses pengembangan sebuah model ontologi dapat dilakukan dengan tahapan sebagai berikut:

- Penentuan domain.
- Penggunaan ulang.
- Penentuan istilah pada ontologi.
- Pendefinisian *class* dan *hierarchy class*.
- Pendefinisian *properties*.

- Pendefinisian *constraints*.
- Tahap pembuatan *instance*.

2.4 RDF

RDF yang memiliki kepanjangan *Resources Description Framework* adalah model sederhana yang menjelaskan relasi antara *properties* dan *values*. Atribut dari sebuah *resource* disebut sebagai *properties*. Atribut tersebut berfungsi untuk merepresentasikan hubungan antar *resource*.

RDF memiliki struktur yang terdiri dari subyek, predikat, dan obyek. Satu struktur tersebut dapat disebut sebagai pernyataan atau kalimat. Subyek adalah sumber daya yang dideskripsikan sebagai URI. Predikat biasanya disebut sebagai properti yang berfungsi untuk menjelaskan keterkaitan subyek dengan obyek. Sedangkan objek memiliki dua tipe berbeda. Contohnya adalah objek yang memiliki tipe URI seperti http://www.id.dbpedia.org/page/Kesultanan_Demak dan objek yang memiliki tipe literal seperti “Perang Ganter”.

Skema yang dimiliki RDF berfungsi untuk mengendalikan sekumpulan terminologi pada dokumen atau bagian kode yang lain. Skema RDF diibaratkan seperti sebuah *master checklist*, atau definisi tata bahasa. Skema ini mampu memodelkan data sederhana dalam bentuk RDF [5].

RDF mengadopsi mekanisme sederhana untuk mengatur berbagai ekspresi. Mekanisme yang digunakan terkait pembuatan *resources*, *properties*, *types* dan *statement* sebagai obyek utama di dalam web. Dengan kata lain, ekspresi tersebut memiliki URI dan tidak mempunyai batasan untuk dikombinasikan satu sama lain. Berikut ini adalah ekspresi dasar RDF [6].

a) Rdfs:resource

Merepresentasikan semua *resource* RDF yang disebut *resources*.

b) Rdfs:Class

Sebuah *resource* dapat dianggap sebagai kelas apabila dibuat sebagai *instance* dari `rdfs:Class`.

- c) `Rdfs:property`
Semua *resource* yang berupa *property* merupakan *instance* dari `rdfs:property`.
- d) `Rdf:type`
Merepresentasikan bahwa sebuah *resource* adalah *instance* sebuah kelas.
- e) `Rdfs:subclassOf`
Merepresentasikan bahwa sebuah kelas adalah sub kelas dari kelas lainnya. Komponen ini mendeskripsikan hubungan *subset* / *superset* antar kelas. Jika terdapat hubungan antara kelas pertama dengan kelas kedua, dan kelas kedua mempunyai hubungan dengan kelas ketiga, maka dapat disimpulkan kelas pertama juga berhubungan dengan kelas ketiga. Oleh karena itu, *subclassOf* disebut sebagai *property* yang bersifat *transitive*. Sub kelas biasanya digunakan oleh satu atau lebih dari satu kelas.
- f) `Rdfs:subPropertyOf`
Merepresentasikan bahwa sebuah *property* adalah sub *property* dari *property* lainnya. Komponen ini memiliki deskripsi hubungan yang sama dengan `rdfs:subclassOf`, yaitu mempunyai hubungan *subset* / *superset* antar kelas dan bersifat *transitive*.
- g) `Rdfs:range`
Merepresentasikan sekumpulan *resource* atau *literal* yang menjadi nilai (*value*) dari sebuah *property*. *Range* dapat diibaratkan sebagai wilayah tertentu dari suatu objek. *Value* yang dapat dimiliki oleh *property* hanya *resource* yang berada pada *range* tertentu. Jika dapat menggunakan semua *resource*, maka *value* akan menjadi tidak terbatas.
- h) `Rdfs:domain`

Merepresentasikan sekumpulan *resource* yang menjadi wilayah subjek dari sebuah *property*. Suatu *property* hanya dapat menggunakan *domain* kelas yang telah didefinisikan. Jika dapat digunakan oleh semua kelas, maka *property* tersebut menjadi tidak terbatas karena bebas digunakan oleh *resource* manapun.

2.5 OWL-Time

OWL-Time adalah sebuah model ontologi yang dikembangkan untuk mendeskripsikan konten temporal pada suatu halaman web dan properti temporal pada *web services*. Ontologi ini menyediakan sebuah kosa kata untuk menyatakan fakta topologi relasi tentang *Instances* dan *Intervals* bersamaan dengan informasi durasi dan informasi tentang hari dan tanggal (*datetime*) [7].

1. Topologi Relasi Temporal

Ada dua sub kelas dari *TemporalEntity* pada skema ontologi OWL-Time yaitu *Instant* dan *Interval*. Topologi kelas *TemporalEntity* dalam OWL ditunjukkan oleh Kode Sumber 2.1.

1	:Instant
2	a owl:Class ;
3	rdfs:subClassOf :TemporalEntity .
4	:Interval
5	a owl:Class ;
6	rdfs:subClassOf :TemporalEntity .
7	:TemporalEntity
8	a owl:Class ;
9	rdfs:subClassOf :TemporalThing ;
10	owl:equivalentClass
11	[a owl:Class ;
12	owl:unionOf (:Instant :Interval)
13] .

Kode Sumber 2.1 Topologi Kelas *TemporalEntity* dalam OWL

Interval adalah sesuatu yang secara intuitif memiliki cakupan sedangkan *Instant* dapat diibaratkan sebuah titik

yang tidak memiliki titik-titik lain di dalamnya. *Instant* dapat diartikan *Interval* yang memiliki panjang nol, dimana titik awal dan akhirnya adalah sama.

Properti *hasBeginning* dan *hasEnds* adalah penghubung antar *Instant* dan *TemporalEntity*. Properti *hasBeginning* mendefinisikan *Instant* yang merupakan titik awal dari suatu *Interval*. Properti *hasEnd* mendefinisikan *Instant* yang merupakan titik akhir dari suatu *Interval*.

2. Relasi Interval

OWL-Time menyediakan properti untuk relasi antar interval, diantaranya *intervalEquals*, *intervalBefore*, *intervalMeets*, *intervalOverlaps*, *intervalStarts*, *intervalDuring*, *intervalFinishes*, dan juga kebalikan (invers) dari relasi-relasi tersebut yaitu *intervalAfter*, *intervalMetBy*, *intervalOverlappedBy*, *intervalStartedBy*, *intervalContains*, *intervalFinishedBy*. Masing-masing dari properti tersebut menghubungkan *instance ProperInterval* dengan *ProperInterval* yang lain.

3. Deskripsi Durasi

Durasi pada sebuah interval dapat memiliki banyak deskripsi. Sebuah interval dapat dideskripsikan sebagai 1 hari 2 jam, atau 26 jam, atau bisa juga dideskripsikan dalam 1560 menit dan seterusnya. Predikat *durationOf* didefinisikan ke dalam 8 argumen, satu untuk *TemporalThing*, dan yang lain adalah tahun, bulan, minggu, hari, jam, menit, dan detik.

1	:DurationDescription
2	a owl:Class ;
3	rdfs:subClassOf
4	[a owl:Restriction ;
5	owl:maxCardinality 1 ;
6	owl:onProperty :seconds
7] ;
8	rdfs:subClassOf
9	[a owl:Restriction ;
10	owl:maxCardinality 1 ;
11	owl:onProperty :minutes

12]	;
13	rdfs:subClassOf	
14	[a owl:Restriction ;
15	owl:maxCardinality 1 ;	
16	owl:onProperty :hours	
17]	
18	rdfs:subClassOf	
19	[a owl:Restriction ;
20	owl:maxCardinality 1 ;	
21	owl:onProperty :days	
22]	
23	rdfs:subClassOf	
24	[a owl:Restriction ;
25	owl:maxCardinality 1 ;	
26	owl:onProperty :weeks	
27]	
28	rdfs:subClassOf	
29	[a owl:Restriction ;
30	owl:maxCardinality 1 ;	
31	owl:onProperty :months	
32]	
33	rdfs:subClassOf	
34	[a owl:Restriction ;
35	owl:maxCardinality 1 ;	
36	owl:onProperty :years	
37]	

Kode Sumber 2.2 Topologi Kelas *DurationDescription* dalam OWL

Satu *interval* dapat memiliki banyak deskripsi durasi namun hanya dapat memiliki satu durasi. *Cardinality* digunakan untuk menentukan *properties/fields* yang harus dan tidak harus dispesifikasikan.

4. Deskripsi *DateTime*

Sebuah *datetime* memiliki properti *unitType*, *year*, *month*, *week*, *day*, *dayOfWeek*, *dayOfYear*, *hour*, *minute*, *second*, dan *timeZone*. Properti *unitType* menspesifikasikan tipe unit temporal dari deskripsi *datetime*. Sebagai contoh, tipe unit dari tanggal 30 adalah hari atau tanggal (*unitDay*). Tipe unit dari 10.40 adalah menit (*unitMinute*), dan seterusnya. Ketika suatu *Instant* memiliki spesifikasi properti *unitType*, maka unit waktu yang lebih kecil dari

yang dispesifikasikan dalam *unitType* tidak dihiraukan. Sebagai contoh, apabila suatu *Instant* memiliki *unitType* bernilai *unitMonth*, maka nilai tanggal, jam, menit dan detik tidak dihiraukan.

OWL-Time mendeskripsikan *datetime* dari suatu *Instant* dengan properti *inDateTime* dengan kelas *DateTimeDescription* sebagai *range*. Masing-masing nilai *datetime* dispesifikasikan ke dalam properti-properti yang sudah disebutkan di atas.

Selain menggunakan *DateTimeDescription*, OWL-Time juga menyediakan metode untuk mendeskripsikan *datetime* dengan lebih sederhana yaitu dengan menggunakan properti *inXSDDatetime* dengan *range* berupa *dateTime* dari skema XML Schema. Metode ini menyimpan *datetime* dengan format *YYY-MM-DDThh-ii-ssZ* dengan keterangan sebagai berikut:

- **YYYY** = 4 digit tahun (*year*)
- **MM** = 2 digit bulan (*month*)
- **DD** = 2 digit tanggal (*day*)
- **hh** = 2 digit jam (*hour*)
- **ii** = 2 digit menit (*minute*)
- **ss** = 2 digit detik (*second*)
- **Z** = Zona waktu

Sebagai contoh, untuk mendeskripsikan *Instant* “Proklamasi” yang memiliki deskripsi *datetime* 10.00 pada tanggal 17 Agustus 1945 dapat diekspresikan dengan menggunakan properti *inXSDDatetime* maupun dengan menggunakan properti *inDateTime*. Kode Sumber 2.3 menunjukkan deskripsi *datetime* dari contoh kasus di atas dalam bahasa OWL.

1	:meetingStart
2	a :Instant ;
3	:inDateTime
4	:meetingStartDescription ;
5	:inXSDDatetime
6	2006-01-01T10:30:00-5:00 .

7	
8	:meetingStartDescription
9	a :DateTimeDescription ;
10	:unitType :unitMinute ;
11	:minute 30 ;
12	:hour 10 ;
13	:day 1 ;
14	:dayOfWeek :Sunday ;
15	:dayOfYear 1 ;
16	:week 1 ;
17	:month 1 ;
18	:timeZone tz-us:EST ;
19	:year 2006 .

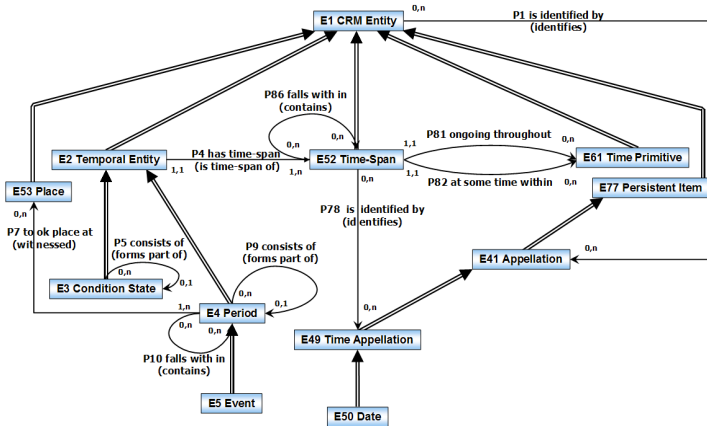
Kode Sumber 2.3 Deskripsi *Datetime* dengan Properti *inXSDDateTime* dan *inDateTime*

2.6 CIDOC-CRM

CIDOC-CRM adalah ontologi yang dapat diperluas untuk konsep dan informasi warisan budaya serta dokumentasi museum. CIDOC merupakan standar internasional (ISO 21127:2014) untuk mengontrol pertukaran informasi warisan budaya. CIDOC adalah sebuah *domain-ontology* yang memiliki *upper-ontology* nya sendiri. Kelas-kelas dari CIDOC melingkupi:

- *Space-Time*: terdiri dari judul, tempat, era/periode, *time-span*, dan hubungan dengan objek-objek persisten.
- *Events*: terdiri dari judul, waktu awal dan akhir, peserta, kreasi atau modifikasi dari suatu bentuk (fisik maupun konsep), dan hubungan dengan objek-objek persisten.
- Objek *Material*: terdiri dari judul, tempat, informasi objek, bagian hubungan, dan hubungan dengan objek-objek persisten.
- Objek *Immaterial*: terdiri dari judul, informasi objek (proporsional maupun simbolik), konsep, dan bagian hubungan.

Gambar 2.1 menunjukkan hirarki hubungan kelas-kelas dari ontologi CIDOC CRM.



Gambar 2.1 Diagram Hirarki Ontologi CIDOC-CRM [8]

2.7 Semantik Web Rule Language (SWRL)

Semantik web rule language adalah suatu bahasa yang menggabungkan antara OWL DL dan OWL Lite, yaitu sub bahasa pada OWL dan Unary atau Binary Datalog RuleML yaitu sub bahasa pada Rule Markup language. Hal ini memungkinkan suatu basis pengetahuan yang dibuat dalam OWL dilengkapi dengan aturan [9].

Aturan dalam SWRL dibangun dari aplikasi antara anteseden (*body*) dan konsekuen (*head*), atau bisa dikatakan bahwa apapun anteseden yang ada, harus ada konsekuen. Anteseden maupun konsekuen harus mengandung minimal nol atau lebih pernyataan. Anteseden kosong berarti konsekuen selalu bernilai true. Sebaliknya, konsekuen kosong berarti konsekuen selalu bernilai false, akibatnya anteseden juga akan bernilai false. Pada Tabel 2.1 dijabarkan definisi atom-atom pada SWRL.

Tabel 2.1 Komponen SWRL

Atom	Deskripsi
C(x)	C adalah deklarasi <i>class</i> dan x adalah nama individual atau variabel
D(y)	D adalah deklarasi <i>data range</i> dan y adalah variabel atau <i>data value</i>
P(x, y)	P adalah deklarasi <i>property</i> P. <i>Class</i> x dan y adalah variabel atau individual. Jika P adalah <i>object property</i> maka y adalah sebuah individual, jika P adalah <i>data property</i> maka y adalah sebuah <i>data value</i>
sameAs(x, y)	x dan y adalah variabel atau individual jika keduanya merupakan individu yang sama. Atom <i>sameAs</i> merupakan negasi dari atom <i>differentFrom</i> .
differentFrom(x, y)	x dan y adalah variabel atau individual jika keduanya merupakan individu yang berbeda Atom <i>differentFrom</i> merupakan negasi dari atom <i>sameAs</i> .

SWRL juga memiliki komponen atom tambahan (built-in) yang bisa digunakan untuk perbandingan, operasi matematik, *boolean*, *string*, *datetime*, URI dan *list*. Dalam tugas akhir ini digunakan beberapa komponen tambahan untuk melakukan perbandingan. Beberapa komponen yang digunakan pada tugas akhir ini dijabarkan dalam sedangkan deskripsi komponen atom secara lengkap dapat dilihat di <https://www.w3.org/Submission/SWRL/#8>.

Tabel 2.2 Komponen SWRL *Built-in*

Atom	Deskripsi
equal(x,y)	x dan y adalah variabel atau individual yang memiliki nilai yang sama. Atom ini digunakan untuk melakukan perbandingan numerik dan <i>datetime</i>
lessThan(x,y)	x dan y adalah variabel atau individual yang memiliki nilai yang sama. Atom ini digunakan untuk melakukan perbandingan numerik dan <i>datetime</i>

Berikut merupakan contoh penggunaan SWRL rule untuk menyatakan bahwa apabila individu x terjadi sebelum(*before*) individu y, maka y terjadi setelah (*after*) x.

```
before(?x, ?y) -> after(?y, ?x)
```

Tanda "?" digunakan untuk mengawali pendeklarasian sebuah variabel atau individual. Tanda "→" berperan sebagai penghubung antara anteseden dan konsekuen. Masing-masing atom dan variabel atau individu dipisahkan oleh tanda ",".

2.8 Google Refine

Google Refine adalah sebuah perangkat lunak untuk mengolah dan memanipulasi data tabel misalnya CSV, TSV dan Excel. Google Refine memungkinkan pengguna untuk memahami data, merapikan dan melakukan transformasi ke dalam format yang diinginkan².

Google Refine tidak mendukung ekspor data dalam format RDF. Digunakan *plugin* RDF Extension untuk pengolahan data RDF. Fungsionalitas ekspor RDF didasarkan

² <http://www.easyrdf.org/>

pada *skeleton* yang memetakan data tabel ke dalam sebuah skema (*graph*). Sistem mengiterasi tiap baris proyek dan menangkap ekspresi Google Refine Expression Language (GREL) pada *skeleton* yang berdasar pada konten kolom untuk menghasilkan sub skema baru.

2.9 EasyRDF

EasyRDF adalah sebuah *library* yang dirancang untuk mempermudah penggunaan dan pembuatan RDF³. *Library* ini telah dirancang untuk penggunaan oleh pengembang RDF berpengalaman maupun pemula. EasyRDF ditulis dalam bahasa pemrograman PHP berbasis obyek.

Dalam penguraian RDF, EasyRDF membangun sebuah grafik objek PHP yang dapat digunakan untuk mendapatkan data untuk ditampilkan pada halaman web.

Data secara khusus dimuat dalam `EasyRdf_Graph` dari kode sumber dokumen RDF. Sumber dokumen dapat berupa file RDF di dalam *web* maupun berupa keluaran dari sebuah query SPARQL *Construct* atau *Describe* dari sebuah *triplestore*.

2.10 SPARQL

Model data RDF berupa suatu *statement* dalam bentuk *triple* yang terdiri dari subjek, predikat, dan objek. Untuk mendapatkan informasi dari suatu graph RDF dibutuhkan suatu *query*. SPARQL adalah bahasa *query* untuk RDF. SPARQL digunakan untuk mengekspresikan *query* yang dibutuhkan dari sebuah *graph* beserta konjungsi dan pemisahannya⁴. *Query* bahasa SPARQL memungkinkan untuk melakukan hal-hal berikut:

1. Mengambil nilai dari data yang terstruktur maupun data yang semi terstruktur.

³ <http://www.easyrdf.org/>

⁴ <https://www.w3.org/TR/rdf-sparql-query/>

2. Mengembangkan data dengan melakukan *query* terhadap suatu relasi yang tidak diketahui.
3. Dapat melakukan *query* operasi *join* yang kompleks pada database yang berlainan secara lebih sederhana.
4. Mengubah suatu data RDF menjadi *vocabulary* yang lain.

Hasil dari *query* SPARQL dapat mengembalikan nilai dalam beberapa *format* data yang antara lain XML, JSON, RDF, dan HTML. Kode Sumber 2.4 adalah contoh sederhana dari SPARQL:

1	PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
2	PREFIX dc: <http://purl.org/dc/elements/1.1/>
3	PREFIX : <http://example/ns#>
4	
5	SELECT ?book ?title
6	WHERE
7	{ ?t rdf:subject ?book .
8	?t rdf:predicate dc:title .
9	?t rdf:object ?title .
10	?t :saidBy "Bob" .
11	}

Kode Sumber 2.4 Contoh *Query* SPARQL

Hasil keluaran dari query di atas dapat dilihat pada Tabel 2.3. Variabel SPARQL dimulai dengan tanda “?” dan merupakan suatu *node* (*resource* atau *literal*) didalam *RDF triple*. Sedangkan pernyataan “SELECT” mengembalikan suatu tabel dari variabel dan nilai yang dideskripsikan didalam query.

Tabel 2.3 Contoh Hasil Keluaran *Query* SPARQL

<i>book</i>	<i>title</i>
<http://example.org/book/book1>	"SPARQL Tutorial"

[Halaman ini sengaja dikosongkan]

BAB III

METODOLOGI PEMECAHAN MASALAH

Pada bab ini dijelaskan mengenai langkah-langkah yang dilakukan dalam pencarian relasi objek-objek warisan budaya. Mulai dari analisis data, perancangan skema ontologi gabungan OWL-Time dan CIDOC-CRM, perancangan *rules* hingga perancangan antarmuka untuk menampilkan data hasil *reasoning*. Data yang dimaksud adalah data yang akan diolah dalam skema ontologi dan kemudian ditampilkan dalam aplikasi sehingga tujuan Tugas Akhir ini bisa tercapai.

3.1 Analisis Data

Pada tugas akhir ini, untuk memecahkan masalah pencarian keterkaitan waktu antar entitas warisan budaya, yang pertama kali dilakukan adalah menganalisis *dataset* yang akan diujikan. Data berupa kejadian, tokoh, benda, dan seluruh elemen sejarah yang memiliki *event* yang terjadi pada suatu satuan waktu. *Event* pada kejadian sejarah misalnya dimulainya Perang Aceh, ditandatanganinya Perjanjian Lingardjati, penobatan sultan pertama Kesultanan Samudra Pasai, dimulainya Invasi Portugis. *Event* pada tokoh sejarah misalnya kelahiran Soekarno, kematian Sutomo, masa pemerintahan Ken Arok. *Event* pada benda sejarah misalnya penemuan keris, eksistensi Kesultanan Mataram, pendirian Monas.

Selain informasi waktu, ontologi yang dirancang juga harus mampu memuat keterangan atau anotasi tentang entitas yang bersangkutan dengan *event*. Keterangan dapat berupa keterangan nama, deskripsi, maupun informasi lainnya yang terkait dengan entitas.

3.1.1 Analisis Data DBpedia

Data yang digunakan dalam tugas akhir ini diambil dari DBpedia Indonesia dan data arkeologi dari Monas. Data arkeologi Monas menyediakan informasi peninggalan benda-

benda bersejarah. DBpedia Indonesia dipilih karena menyediakan informasi terstruktur dari Wikipedia Indonesia. Sebagai contoh, pada halaman *resource* DBpedia Kesultanan Aceh yang diakses melalui http://id.dbpedia.org/page/Kesultanan_Aceh, ditampilkan tabel yang berisi data-data yang berkaitan dengan Kesultanan Aceh berdasarkan ontologi yang dipakai oleh DBpedia seperti yang ditunjukkan oleh Gambar 3.1.



Property	Value
dbpprop-id:commonName	Kesultanan Aceh
dbpprop-id:continent	Asia
dbpprop-id:conventionalLongName	Kesultanan Aceh Darussalam
dbpprop-id:country	Indonesia
dbpprop-id:currency	Koin emas dan perak lokal
dbpprop-id:eventEnd	dbpedia-id:Perang_Aceh
dbpprop-id:eventStart	Penobatan sultan pertama
dbpprop-id:flagS	Flag of the Netherlands.svg
dbpprop-id:governmentType	Monarki absolut
dbpprop-id:imageCoat	Cap Sikureung.jpg
dbpprop-id:imageFlag	Bendera aceh.svg
dbpprop-id:imageMap	Aceh Sultanate id.svg
dbpprop-id:imageMapCaption	Luas Kesultanan Aceh pada masa pemerintahan Sultan Iskandar Muda. 1608-1637.
dbpprop-id:leader	<ul style="list-style-type: none"> Ali Mughayat Syah Alaadin Muhammad Daud Syah
dbpprop-id:p	Lamuri
dbpprop-id:region	Asia Tenggara

Gambar 3.1 Halaman *Resource* DBpedia

3.1.2 Analisis Granularitas

Anotasi waktu yang terdapat pada *resource* DBpedia direpresentasikan sebagai satuan waktu primitif dan non primitif. Satuan waktu primitif ditampilkan sebagai tanggal, bulan, tahun, dan abad. Tanggal, bulan dan tahun disimpan dalam format *dateTime*, *date*, *gYear*, *gDate*, *gMonth*, dan sebagainya dalam XML Schema. Informasi dalam satuan abad biasanya ditampilkan dalam format teks karena tidak ada skema penyimpanan yang dapat menyimpan granularitas pada *level* abad.

Waktu bernilai non primitif pada *resource* DBpedia ditampilkan dalam teks seperti pada informasi abad. Sebagai contoh seperti yang ditunjukkan pada Gambar 3.1, Kerajaan Singhasari memiliki properti *eventStart* Perang Gantar.

3.2 Ekstraksi Data

Ekstraksi dilakukan untuk mengambil semua data yang terkandung dalam *resource* DBpedia yang diperlukan untuk pengujian dalam tugas akhir ini.

3.2.1 Ekstraksi Data Menggunakan SPARQL Endpoint

DBpedia menyediakan SPARQL *endpoint* yang digunakan untuk ekstraksi data menggunakan *query* SPARQL. SPARQL *endpoint* DBpedia Indonesia dapat diakses melalui URL <http://id.dbpedia.org/sparql>. SPARQL *endpoint* DBpedia Indonesia menggunakan Virtuoso SPARQL *Query Editor* seperti yang ditunjukkan oleh Gambar 3.2.

Virtuoso SPARQL Query Editor

Default Data Set Name (Graph IRI)

Query Text

```

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbp: <http://dbpedia.org/ontology/>
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX time: <http://www.w3.org/2006/time#>
PREFIX dbp: <http://dbpedia.org/property/>

select distinct ?entity ?date
where
{
  {
    dbr:Sukarno ?property ?date .
    dbr:Sukarno rdfs:label ?subject .
    ?property rdf:type rdf:Property .
    FILTER ( CONTAINS( str(?property), "Date" ) ||
             CONTAINS( str(?property), "date" ) ).
  }
}

```

Sporing: Use only local data (including data retrieved before), but do not retrieve more

Results Format: HTML

Execution timeout: 0 milliseconds (values less than 1000 are ignored)

Options: ☒ Strict checking of void variables

(The result can only be sent back to browser, not saved on the server, see [details](#))

Run Query Reset

Copyright © 2016 OpenLink Software
Virtuoso version 06.01.3127 on Linux (64-bit linux-gnu), Single Server Edition

Gambar 3.2 Virtuoso SPARQL Query Editor DBpedia

Virtuoso menerima masukan berupa SPARQL *query* dan *Default Data Set Name* (Graph IRI). Sedangkan keluarannya dapat berupa tabel HTML dan file dengan format NTriples dan beberapa pilihan lainnya seperti RDF, XML, dan JSON, namun hanya tabel HTML dan Ntriples yang dapat digunakan.

Untuk mendapatkan data dari *resource*, dibutuhkan *query* SPARQL yang berfungsi mengambil *property* dan *value* yang dibutuhkan dari *resource* tersebut. Kode Sumber 3.1 adalah *query* untuk mendapatkan properti dan *value* dari sebuah *resource* DBpedia. Properti yang diambil memiliki beberapa kondisi. Kondisi pertama adalah ketika properti tersebut mengandung kata “date” atau “Date”. Kondisi kedua, ketiga, dan keempat adalah ketika properti tersebut memiliki *range* yang berformat *dateTime*, *date*, *gYear*. Keempat kondisi tersebut kemudian digabungkan dengan menggunakan fungsi *union*.

1	PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
2	PREFIX owl: <http://www.w3.org/2002/07/owl#>
3	PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
4	PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
5	PREFIX dbo: <http://dbpedia.org/ontology/>
6	PREFIX dbr: <http://dbpedia.org/resource/>
7	PREFIX time:<http://www.w3.org/2006/time#>
8	PREFIX dbp: <http://dbpedia.org/property/>
9	
10	select distinct ?entity ?property ?date
11	where
12	{
13	{
14	dbr:Sukarno ?property ?date .
15	dbr:Sukarno rdfs:label ?subject .
16	?property rdf:type rdf:Property .
17	FILTER (CONTAINS(str(?property), "Date")
18	CONTAINS(str(?property), "date")).
19	}

20	union
21	{
22	dbr:Sukarno ?property ?date .
23	dbr:Sukarno rdfs:label ?subject .
24	?property rdf:type rdf:Property ;
25	rdfs:range xsd:dateTime .
26	}
27	union
28	{
29	dbr:Sukarno ?property ?date .
30	dbr:Sukarno rdfs:label ?subject .
31	?property rdf:type rdf:Property ;
32	rdfs:range xsd:date .
33	}
34	union
35	{
36	dbr:Sukarno ?property ?date .
37	dbr:Sukarno rdfs:label ?subject .
38	?property rdf:type rdf:Property ;
39	rdfs:range xsd:gYear .
40	}
41	FILTER (LANG(?subject) = 'en') .
42	BIND (SUBSTR(str(?property),29) AS ?p) .
43	BIND (CONCAT (str(?subject),"_", str(?p)) AS ?entity)
44	}
45	limit 100

Kode Sumber 3.1 SPARQL Query untuk Ekstraksi Data DBpedia

Query tersebut diolah oleh Virtuoso SPARQL Editor dan hasilnya ditampilkan melalui halaman HTML dalam bentuk tabel. Hasil *query* SPARQL ekstraksi data dengan Kode Sumber 3.1 ditunjukkan oleh Tabel 3.1.

Tabel 3.1 Hasil SPARQL Query Ekstraksi Data DBpedia

<i>entity</i>	<i>property</i>	<i>date</i>
Sukarno_deathDate	http://dbpedia.org/ontology/deathDate	1970-06-21
Sukarno_birthDate	http://dbpedia.org/ontology/birthDate	1901-06-06
Sukarno_birthDate	http://dbpedia.org/property/birthDate	1901-06-06
Sukarno_dateOfBirth	http://dbpedia.org/property/dateOfBirth	1901-06-06
Sukarno_dateOfDeath	http://dbpedia.org/property/dateOfDeath	1970-06-21
Sukarno_deathDate	http://dbpedia.org/property/deathDate	1970-06-21
Sukarno_activeYearsEndDate	http://dbpedia.org/ontology/activeYearsEndDate	1966-07-25
Sukarno_activeYearsEndDate	http://dbpedia.org/ontology/activeYearsEndDate	1967-03-12
Sukarno_activeYearsStartDate	http://dbpedia.org/ontology/activeYearsStartDate	1945-08-18
Sukarno_activeYearsStartDate	http://dbpedia.org/ontology/activeYearsStartDate	1959-07-09
Sukarno_birthYear	http://dbpedia.org/ontology/birthYear	"1901"^^< http://www.w3.org/2001/XMLSchema#gYear >

<i>entity</i>	<i>property</i>	<i>date</i>
Sukarno_deathYear	<i>http://dbpedia.org/ontology/deathYear</i>	"1970"^^<http://www.w3.org/2001/XMLSchema#gYear>

Tabel 3.1 berhasil menampilkan seluruh data yang dibutuhkan dari *resource* Sukarno sesuai dengan kondisi-kondisi yang didefinisikan dalam Kode Sumber 3.1. Kode tersebut juga dapat berjalan pada *resource* yang memiliki properti-properti yang identik dengan *resource* Sukarno.

Ekstraksi data menggunakan SPARQL *query* menemui kesulitan ketika ditemukan fakta bahwa tipe data yang digunakan untuk penyimpanan informasi waktu pada *resource* DBpedia tidak tetap. Tidak ada keseragaman tipe data untuk penyimpanan informasi waktu. Sebagai contoh, pada *resource* Kesultanan_Aceh, penyimpanan informasi tahun menggunakan tipe data *integer*. Jelas berbeda apabila dibandingkan dengan *resource* Sukarno yang menggunakan tipe data *gYear* dan *date*. Pada *resource* yang sama juga ditemukan penyimpanan informasi waktu dengan menggunakan teks pada abstrak maupun properti lain. Ada juga yang menggunakan URI yang merujuk ke *resource* lain sebagai *value* dalam penyimpanan informasi waktu. Tak jarang pula informasi waktu ditampilkan pada paragraf abstrak. Halaman *resource* Kesultanan_Aceh ditunjukkan oleh Gambar 3.3

dbpprop-id:eventEnd	▪ dbpedia-id:Perang_Aceh
dbpprop-id:eventStart	▪ Penobatan sultan pertama
dbpprop-id:flagS	▪ Flag of the Netherlands.svg
dbpprop-id:governmentType	▪ Monarki absolut
dbpprop-id:imageCoat	▪ Cap Sikureueng.jpg
dbpprop-id:imageFlag	▪ Bendera aceh.svg
dbpprop-id:imageMap	▪ Aceh Sultanate id.svg
dbpprop-id:imageMapCaption	▪ Luas Kesultanan Aceh pada masa pemerintahan Sultan Iskandar Muda, 1608-1637.
dbpprop-id:leader	▪ Ali Mughayat Syah
	▪ Alaidin Muhammad Daud Syah
dbpprop-id:p	▪ Lamuri
dbpprop-id:region	▪ Asia Tenggara
dbpprop-id:religion	▪ dbpedia-id:Islam
dbpprop-id:s	▪ Hindia Belanda
dbpprop-id:symbolType	▪ Cap Sikureueng
dbpprop-id:titleLeader	▪ Sultan
dbpprop-id:yearEnd	▪ 1903 (xsd:integer)
dbpprop-id:yearLeader	▪ 1496 (xsd:integer)
	▪ 1874 (xsd:integer)
dbpprop-id:yearStart	▪ 1496 (xsd:integer)

Gambar 3.3 Penyimpanan Semantik Informasi Waktu pada DBpedia

Selain ketidakpastian tipe data, properti yang digunakan dalam menyampaikan informasi waktu juga bermacam-macam. Pada *resource* Kesultanan_Aceh yang ditunjukkan oleh Gambar 3.3 ditemukan properti *eventStart* dan *eventEnd*. Properti ini jelas tidak tertangkap oleh query karena tidak masuk dalam satu pun kondisi yang didefinisikan.

DBpedia sebenarnya memiliki *mapping* berdasarkan region. DBpedia Indonesia sendiri memiliki *mapping* yang dapat diakses melalui http://mappings.dbpedia.org/index.php/Mapping_id. Akan tetapi pada kenyataannya, ontologi yang diterapkan untuk menampilkan informasi pada halaman resource DBpedia Indonesia bukan hanya kelas-kelas dan properti dari *mapping* DBpedia Indonesia saja melainkan keseluruhan kelas dan properti yang ada pada ontologi DBpedia.

Dalam ekstraksi data DBpedia menggunakan SPARQL *query*, agar dapat mengambil yang dibutuhkan secara keseluruhan harus dibuat *query* baru yang mampu melingkupi semua properti dan *value*-nya yang memungkinkan untuk menyimpan informasi waktu. Hal ini dianggap tidak memungkinkan karena luasnya ontologi yang dimiliki oleh DBpedia. Atas alasan-alasan itulah metode ekstraksi

menggunakan SPARQL query tidak digunakan dalam tugas akhir ini.

3.2.2 Ekstraksi Data Manual

Ekstraksi manual adalah metode ekstraksi dengan cara membaca langsung konten resource DBpedia, kemudian memasukkannya ke dalam tabel.

3.2.2.1 Ekstraksi Tahap 1

Informasi-informasi terkait *resource* pada DBpedia ditampilkan dalam bentuk tabel *property* dan *value*. Properti ditampilkan menurut properti yang ada pada skema ontologi yang digunakan oleh DBpedia sedangkan *value* ditampilkan dalam format tertentu, baik literal maupun URI.

Pada tahap ini data-data dari *resource* diseleksi menurut kebutuhan dalam studi kasus ini dan kemudian dimasukkan ke dalam tabel. Hasil ekstraksi tahap 1 ditunjukkan oleh Tabel 3.2 Hasil Ekstraksi Data Tahap 1.

Tabel 3.2 Hasil Ekstraksi Data Tahap 1

<i>resource</i>	Kesultanan_Aceh
<i>commonName</i>	Kesultanan Aceh
<i>conventionalLongName</i>	Kesultanan Aceh Darussalam
<i>country</i>	Indonesia
<i>region</i>	Asia Tenggara
<i>eventEnd</i>	Perang_Aceh
<i>eventStart</i>	Penobatan sultan pertama
<i>leader</i>	Ali Mughayat Syah
<i>leader</i>	Alaidin Muhammad Daud Syah
<i>yearEnd</i>	1903
<i>yearStart</i>	1496
<i>yearLeader</i>	1496
<i>yearLeader</i>	1874

<i>label</i>	Kesultanan Aceh
<i>comment</i>	Kesultanan Aceh Darussalam merupakan sebuah kerajaan Islam yang pernah berdiri di provinsi Aceh, Indonesia. Kesultanan Aceh terletak di utara pulau Sumatera dengan ibu kota Bandar Aceh Darussalam dengan sultan pertamanya adalah Sultan Ali Mughayat Syah yang dinobatkan pada Ahad, 1 Jumadil awal 913 H atau pada tanggal 8 September 1507.

3.2.2.2 Ekstraksi Tahap 2

Setelah dilakukan Ekstraksi tahap 1 yang dilakukan selanjutnya adalah memilah kembali data hasil ekstraksi tahap 1 yang memiliki potensi untuk dijadikan suatu “*event*”. *Event* dibentuk dari suatu entitas yang memiliki keterangan waktu baik yang didefinisikan dalam bentuk tanggal dan waktu maupun yang didefinisikan dalam bentuk *event* yang lain. Dilihat dari hasil ekstraksi tahap 1, entitas yang bisa dijadikan suatu *event* adalah Kesultanan_Aceh, Ali Mughayat Syah dan Alaidin Muhammad Daud Syah. Hasil dari ekstraksi tahap 2 dapat dilihat pada Tabel 3.3.

Tabel 3.3 Hasil Ekstraksi Data Event dari Resource DBpedia Tahap 2

Kode Event	Properti	Value
Event 001	<i>Entity</i>	Kesultanan_Aceh
	<i>Event</i>	Era Kesultanan_Aceh
	<i>Start</i>	Penobatan sultan pertama
	<i>End</i>	Perang_Aceh
	<i>Start value</i>	1496
	<i>End value</i>	1903
	<i>Entity</i>	Ali Mughayat Syah

Event 002	<i>Event</i>	Era Ali Mughayat Syah
	<i>Start</i>	Awal Era Ali Mughayat Syah
	<i>End</i>	Akhir Era Ali Mughayat Syah
	<i>Start value</i>	1496
	<i>End value</i>	1874
Event 003	<i>Entity</i>	Alaidin Muhammad Daud Syah
	<i>Event</i>	Era Alaidin Muhammad Daud Syah
	<i>Start</i>	Awal Era Alaidin Muhammad Daud Syah
	<i>End</i>	Akhir Era Alaidin Muhammad Daud Syah
	<i>Start value</i>	1874
	<i>End value</i>	1903

Pada Tabel 3.3, penambahan dan modifikasi *value* diterapkan untuk menyesuaikan kebutuhan pada skema ontologi yang digunakan. Skema ontologi membutuhkan *instances* yang mendefinisikan titik awal dan titik akhir. Penambahan ini juga didasarkan pada penalaran dan dengan bantuan deskripsi yang ditampilkan pada halaman *resource* DBpedia. Deskripsi yang dimaksud umumnya terletak di bagian atas halaman atau bisa juga didefinisikan dalam properti tertentu. Properti yang paling sering digunakan untuk memuat deskripsi adalah properti *abstract*.

Pada kasus *resource* Kesultanan_Aceh, tabel *property-value* mendefinisikan dua *yearLeader* tanpa diketahui siapa *leader* pertama dan *leader* kedua. Namun masalah tersebut terpecahkan melalui keterangan yang dimuat dalam properti *abstract*. Disebutkan sultan pertama Kesultanan Aceh adalah Sultan Ali Mughayat Syah. Maka dapat disimpulkan bahwa sultan kedua adalah Alaidin Muhammad Daud Syah. Dengan demikian, *yearLeader* dengan nilai yang lebih besar adalah milik Alaidin Muhammad Daud Syah dan dengan penalaran didapatkan bahwa tahun 1874 menandakan awal era kepemimpinan Alaidin Muhammad Daud Syah.

Selain informasi waktu, ekstraksi tahap 2 menyiasakan informasi lain yaitu informasi tempat dan keterangan terkait entitas yang dipilih seperti yang ditunjukkan oleh Tabel 3.4

Tabel 3.4 Hasil Ekstraksi Data Anotasi dari ResourceDBpedia

Entitas	Properti	Value
Kesultanan_Aceh	commonName	Kesultanan Aceh
	conventionalLong Name	Kesultanan Aceh Darussalam
	country	Indonesia
	region	Asia Tenggara
	label	Kesultanan Aceh
	comment	Kesultanan Aceh Darussalam merupakan sebuah kerajaan Islam yang pernah berdiri di provinsi Aceh, Indonesia. Kesultanan Aceh terletak di utara pulau Sumatera dengan ibu kota Bandar Aceh Darussalam dengan sultan pertamanya adalah Sultan Ali Mughayat Syah yang dinobatkan pada pada Ahad, 1 Jumadil awal 913 H atau pada tanggal 8 September 1507.

Ekstraksi dilakukan pada masing-masing data dari halaman *resource* DBpedia yang akan digunakan untuk pengujian yakni 23 *event*.

3.2.3 Transformasi Data Menggunakan Google Refine

Pada tahap ini dijelaskan mengenai proses transformasi data ke dalam format rdf agar dapat diproses dengan menggunakan skema ontologi yang dibangun. Transformasi data dilakukan dengan menggunakan *tool* Google Refine 2.5 menggunakan bantuan ekstensi RDF extension 0.8.

Yang dilakukan pertama kali adalah memasukkan data ke dalam satu tabel excel. Data-data dikelompokkan ke dalam kolom-kolom tabel berdasarkan tipe (kelas). Data yang berada dalam baris yang sama adalah data yang saling berhubungan. Contoh *record* data untuk masukan Google Refine ditunjukkan oleh Tabel 3.5.

Tabel 3.5 Contoh *Record* untuk Masukan Google Refine

Interval	Instant Awal	Nilai Instant Awal	Instant Akhir	Nilai Instant Akhir
001_PI_Era_Kerajaan_Singhasari	001_I_Awal_Era_Kerajaan_Singhasari	1222-01-01T00:00:00	001_I_Akhir_Era_Kerajaan_Singhasari	1292-12-31T00:00:00

Tabel excel yang terbentuk diimpor ke dalam proyek baru pada Open Refine. Setelah berhasil diimpor kemudian dibuat rancangan *skeleton* atau skema yang menghubungkan antar kolom dalam tabel. Skema ontologi yang diimpor adalah ontologi yang dipakai dalam studi kasus ini, yaitu OWL-Time, CIDOC-CRM, dan XSD. Hal ini bertujuan untuk mempermudah perancangan *skeleton*. Penambahan ontologi dilakukan dengan menambahkan *prefix* ontologi. Daftar *prefix* yang diimpor untuk perancangan *skeleton* ditunjukkan oleh Tabel 3.6.

Tabel 3.6 Daftar *Prefix* yang Digunakan untuk Perancangan *Skeleton*

Prefix	URI
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs	http://www.w3.org/2000/01/rdf-schema#
time	http://www.w3.org/2006/time#
cidoc-crm	http://www.cidoc-crm.org/cidoc-crm/

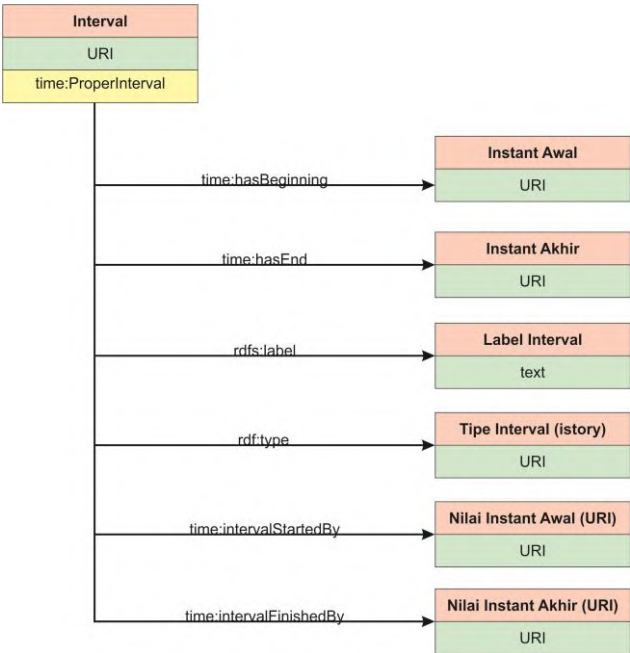
Tahap selanjutnya adalah menambahkan *base* URI. *Base* URI ini harus sama dengan URI yang dipakai dalam rancangan ontologi yaitu <http://www.istory.id#>. *Base* URI ini nantinya akan menjadi URI dari individu-individu baru hasil ekspor Google Refine.

Selanjutnya kolom-kolom dari tabel ditransformasikan sebagai *node-node* dalam skeleton. Google Refine membaca *table head* atau kepala tabel sebagai *identifier* untuk masing-masing kolom sehingga penambahan *node* mengacu pada *table head*. Kolom-kolom inti dari tabel diinisiasikan sebagai *root* dan kolom lain yang berhubungan dihubungkan dengan *root* menggunakan sebuah properti. Properti-properti ini didapat melalui hasil impor skema ontologi.

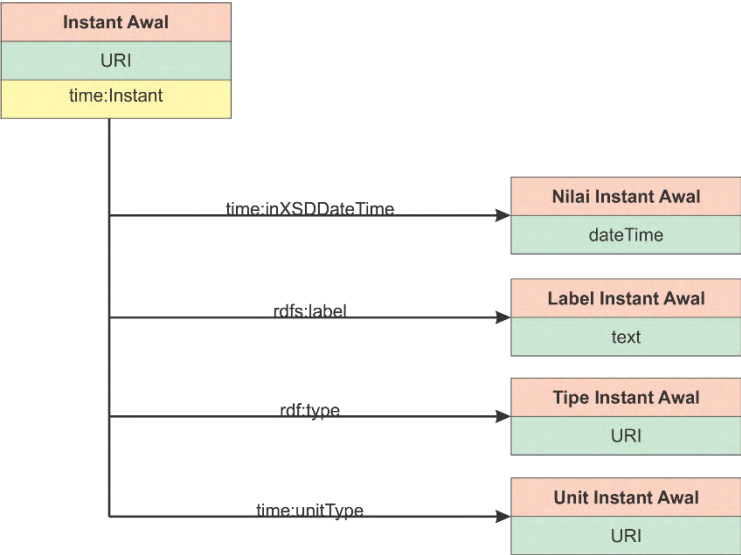
Tipe dari masing-masing *node* ditambahkan dengan cara yang sama dengan proses yang sama dengan penambahan properti yaitu dengan menambahkan *prefix* dan tipe yang diinginkan melalui fitur *add rdf:type*. Rancangan *skeleton* ditunjukkan oleh Gambar 3.4, Gambar 3.5, Gambar 3.6 dan Gambar 3.7.



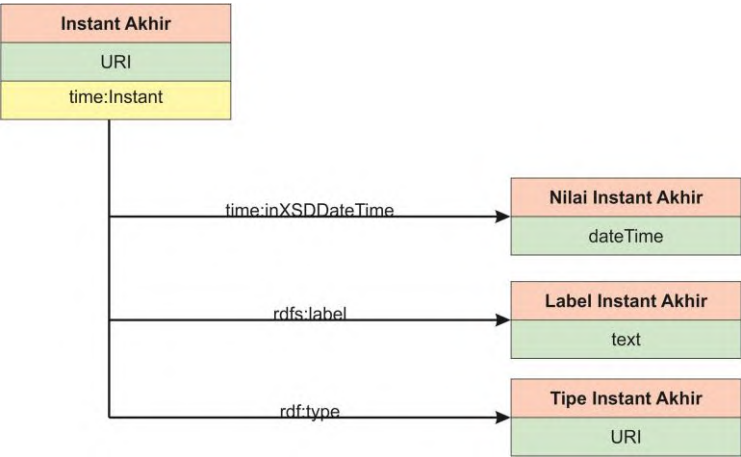
Gambar 3.4 Rancangan *Skeleton Root Entitas*



Gambar 3.5 Rancangan *Skeleton Root Interval*



Gambar 3.6 Rancangan *Skeleton Root Instant Awal*



Gambar 3.7 Rancangan *Skeleton Root Instant Akhir*

Record dalam tabel akan otomatis terpetakan berdasarkan rancangan *skeleton* yang telah dibuat. Transformasi tabel ke dalam *skeleton* menghasilkan individu-individu baru dalam bentuk RDF. Hasil transformasi kemudian diekspor dalam format RDF/XML. Contoh hasil transformasi tabel ke dalam bentuk RDF ditampilkan pada Kode Sumber 3.2.

1	<rdf:Description rdf:about="http://www.istory.id#001_PI_Era_Kerajaan_Singhasari">
2	<rdf:type rdf:resource="http://www.w3.org/2006/time#ProperInterval"/>
3	</rdf:Description>

Kode Sumber 3.2 Contoh Hasil Transformasi Tabel Ke Dalam Format RDF/XML

Seluruh individu hasil tranformasi tabel disalin ke dalam ontologi yang dibuat sehingga terbentuk individu-individu baru yang bisa diolah untuk dicari relasinya.

Kelas-kelas dan properti tambahan yang berasal dari ontologi CIDOC-CRM ditambahkan secara manual melalui aplikasi Protégé. Hal ini dikarenakan Google Refine tidak dapat mengimpor ontologi CIDOC-CRM.

3.3 Perancangan Ontologi

Pada tahap ini skema ontologi dibangun untuk menyimpan informasi yang telah didapat. Dalam tugas akhir ini ontologi yang dipakai adalah gabungan dari OWL-Time dan CIDOC-CRM. OWL-Time mendefinisikan skema pengolahan data waktu karena memiliki skema penyimpanan waktu yang bisa digunakan untuk menyimpan informasi waktu baik secara angka dalam format *xsd:dateTime* maupun bukan angka dalam URI. CIDOC-CRM digunakan untuk penyimpanan deskripsi

seperti jenis, nama, tempat, dan informasi lainnya yang melekat pada suatu entitas.

Skema ontologi pada tugas akhir ini dibangun dari skema ontologi yang sudah ada. Pada awalnya skema yang akan digunakan hanyalah OWL-Time. Pemilihan skema ontologi OWL-Time didasarkan oleh kemampuannya menyimpan informasi waktu dalam format *xsd:dateTime* sehingga pencarian relasi satu *temporal entity* terhadap *temporal entity* yang lain dapat dilakukan dengan menggunakan fitur yang dimiliki SWRL. Skema ontologi dari OWL-Time dapat dimuat dari URL <http://www.w3.org/2006/time>. Beberapa kelas dan properti ditambahkan untuk memudahkan pencarian relasi antar *temporal entity*. Penambahan ini berdasarkan konsep ontologi CHRONOS [10] yang dapat dimuat dari URL <http://www.intelligence.tuc.gr/2011/timeEvents>.

Pada tahap analisis data didapatkan bahwa suatu instance memiliki informasi lain yang melekat seperti tipe, jenis, nama, tempat, dan informasi lainnya. Maka dibutuhkan sebuah skema penyimpanan yang mampu menyimpan detail informasi tersebut. Alasan tersebutlah yang mendasari digunakannya skema ontologi CIDOC-CRM pada tugas akhir ini. Skema CIDOC-CRM memiliki komponen kelas yang sesuai untuk menyimpan deskripsi *instance*. CIDOC-CRM yang digunakan dalam tugas akhir ini adalah v.5.0.4 yang dapat dimuat dari URL http://www.cidoc-crm.org/rdfs/cidoc_crm_v5.0.4_official_release.rdfs.

3.3.1 Skema Penyimpanan Informasi Temporal

Data *temporal entity* memiliki informasi temporal yang didefinisikan dalam bentuk angka seperti tanggal, bulan, dan tahun maupun dalam bentuk *temporal entity* lain. Skema ontologi CIDOC-CRM memiliki skema penyimpanan informasi waktu dalam URI sebagai individu baru. Skema ini tidak memungkinkan untuk dijadikan penyimpanan informasi tanggal, bulan, dan tahun dalam angka. Padahal dalam studi

kasus pencarian relasi waktu objek-objek antar warisan budaya ini membutuhkan penyimpanan skema penyimpanan dalam bentuk angka yang dapat diolah secara matematis untuk memudahkan pencarian.

Skema OWL-Time memiliki penyimpanan informasi waktu dalam bentuk *xsd:dateTime*. Bentuk penyimpanan ini memungkinkan untuk dilakukan operasi perbandingan secara matematis dengan menggunakan *SWRL rule*. *Format* yang digunakan oleh *xsd:dateTime* adalah YYYY-MM-DDThh-mm-ss yang mengacu pada ISO8601 [11] dimana YYYY adalah 4 digit tahun, MM adalah 2 digit bulan, DD adalah 2 digit tanggal, hh adalah 2 digit jam, mm adalah 2 digit menit, dan ss adalah 2 digit detik. OWL-Time juga memungkinkan menyimpan informasi waktu sebagai URI. Dengan demikian dapat disimpulkan bahwa skema OWL-Time cocok untuk menyimpan informasi waktu yang melekat pada objek-objek warisan budaya.

Beberapa objek warisan budaya pada DBpedia mencantumkan informasi waktu hanya sebagian saja dari standar *format* yang diterima oleh skema *xsd:dateTime*. Gambar 3.8 menunjukkan bahwa informasi waktu yang diketahui pada *resource* DBpedia hanyalah informasi tahun saja. Tentu ini tidak sesuai dengan *format* yang ditangkap oleh skema ontologi. Dibutuhkan keterangan waktu tambahan agar informasi yang dimuat dalam *resource* DBpedia dapat disimpan dalam format *xsd:dateTime*.

dbpprop-id:tahun	▪ 1222 (xsd:integer)
dbpprop-id:titleLeader	▪ Raja
dbpprop-id:yearEnd	▪ 1292 (xsd:integer)
dbpprop-id:yearLeader	▪ 1222 (xsd:integer) ▪ 1268 (xsd:integer)
dbpprop-id:yearStart	▪ 1222 (xsd:integer)

Gambar 3.8 Penyimpanan Informasi Waktu pada DBpedia

Dalam penerapan studi kasus ini, penambahan dilakukan untuk melengkapi data yang disediakan DBpedia agar dapat ditangkap oleh skema *xsd:dateTime*. Titik awal sebuah interval yang hanya diketahui tahunnya diasumsikan memiliki bulan, tanggal, jam, menit dan detik paling awal dari tahun tersebut sedangkan titik akhir diasumsikan memiliki bulan, tanggal, jam, menit dan detik paling akhir dari tahun tersebut. Ini juga berlaku pada abad. Sebagai contoh, properti *yearStart* yang pada DBpedia bernilai 1222 seperti yang ditunjukkan oleh Gambar 3.8 disimpan dengan nilai 1222-01-01T00:00:00. Properti *yearEnd* yang pada DBpedia bernilai 1292 disimpan dengan nilai 1292-12-31T23:59:59. Dengan demikian data waktu dapat ditangkap oleh format penyimpanan *xsd:dateTime*.

Data yang ditambahkan untuk melengkapi data asli bukanlah data yang sebenarnya dan tidak tertera pada *resource* DBpedia. Dalam kasus ini digunakan properti *unitType* untuk mengetahui satuan waktu terkecil sebenarnya yang melekat pada suatu *Instant*. Nilai dari properti *unitType* adalah salah satu dari *unitSecond*, *unitMinute*, *unitHour*, *unitDay*, *unitWeek*, *unitMonth* dan *unitYear*. Properti *unitType* menspesifikasikan tipe unit *temporal* dari deskripsi *datetime* (*DateTimeDescription*). Dengan menambahkan properti *unitType*, data waktu yang sebenarnya dapat ditampilkan pada sisi aplikasi.

Masing-masing entitas temporal didefinisikan sebagai interval sebagaimana yang dijelaskan pada sub bab OWL-Time

dan disimpan dalam kelas *ProperInterval*. Setiap entitas temporal memiliki titik awal dan titik akhir.

Dalam beberapa kasus warisan budaya, sering ditemukan inkonsistensi waktu terkait perbedaan sistem penanggalan maupun faktor lainnya. Hal ini bisa menimbulkan makna ganda yang menyebabkan kesalahan dalam pencatatan waktu pada objek-objek warisan budaya. Menurut teori Open-World Assumption, bahasa yang digunakan oleh web semantik termasuk OWL tidak memberi kesimpulan tanpa adanya pengetahuan yang mendasarinya. Sehingga pada kasus ini tetap dibutuhkan adanya campur tangan manusia untuk memutuskan mana yang benar.

3.3.2 Seleksi Kelas dan Properti

CIDOC-CRM adalah model ontologi yang dikhususkan untuk memodelkan penyimpanan anotasi warisan budaya. Kelas-kelas dan properti bawaan dari CIDOC-CRM mampu menampung anotasi yang melekat pada benda bersejarah, tokoh maupun *event* atau kejadian sejarah. Dalam studi kasus ini, tidak semua anotasi yang melekat pada *instance* disertakan.

Pada OWL-Time, dikarenakan skema yang digunakan untuk memodelkan penyimpanan waktu adalah menggunakan skema *xsd:datetime*, maka kelas-kelas dan properti yang digunakan adalah yang berhubungan dengan skema *xsd:datetime* saja. Kelas-kelas yang berhubungan dengan skema durasi dan deskripsi *datetime* juga tidak digunakan karena tidak termasuk dalam studi kasus ini. Deskripsi *datetime* hanya sebagai *inferred data range* dari inisiasi *temporal unit*. Penggunaan kelas ditunjukkan oleh Tabel 3.7, penggunaan *object property* ditunjukkan oleh Tabel 3.8 dan penggunaan *data property* ditunjukkan oleh Tabel 3.9.

Tabel 3.7 Penggunaan *Classes* dari OWL-Time dan CIDOC-CRM

<i>Classes</i>		
<i>ProperInterval</i>	URI	http://www.w3.org/2006/time#ProperInterval
	Fungsi	Mendekripsikan <i>instance</i> sebagai waktu <i>interval</i>
<i>Instant</i>	URI	http://www.w3.org/2006/time#Instant
	Fungsi	Mendekripsikan <i>instance</i> sebagai waktu titik (<i>instant</i>)
<i>TemporalUnit</i>	URI	http://www.w3.org/2006/time#TemporalUnit
	Fungsi	Mendekripsikan <i>instance</i> sebagai <i>temporal unit</i> .
<i>Event</i> (dan <i>child</i> -nya)	URI	http://www.intelligence.tuc.gr/2011/timeEvents#Event http://www.cidoc-crm.org/cidoc-crm/E5_Event
	Fungsi	Mendekripsikan <i>instance</i> sebagai kejadian/ peristiwa.
<i>Pesistent Item</i> (dan <i>child</i> -nya)	URI	http://www.cidoc-crm.org/cidoc-crm/E77_Persistent_Item
	Fungsi	Mendekripsikan <i>instance</i> sebagai objek persisten.

Tabel 3.8 Penggunaan *Object Property* dari OWL-Time dan CIDOC-CRM

<i>Object Properties</i>		
<i>before</i>	URI	http://www.w3.org/2006/time#before
	Fungsi	Mendeskrripsikan kejadian sebelum dari suatu <i>ProperInterval</i> atau <i>Instant</i> .
<i>after</i>	URI	http://www.w3.org/2006/time#after
	Fungsi	Mendeskrripsikan kejadian setelah dari suatu <i>ProperInterval</i> atau <i>Instant</i> .

<i>Object Properties</i>		
<i>equals</i>	URI	http://www.intelligence.tuc.gr/2011/timeEvents#equals
	Fungsi	Mendeskripsikan kejadian yang bersamaan dari suatu <i>Instant</i> .
<i>intervalMeets</i>	URI	http://www.w3.org/2006/time#intervalMeets
	Fungsi	Mendeskripsikan <i>instance</i> sebagai <i>ProperInterval</i> yang bertemu dengan <i>ProperInterval</i> lain.
<i>intervalMetBy</i>	URI	http://www.w3.org/2006/time#intervalMetBy
	Fungsi	Mendeskripsikan <i>instance</i> sebagai <i>ProperInterval</i> yang ditemui oleh <i>ProperInterval</i> lain.
<i>intervalBefore</i>	URI	http://www.w3.org/2006/time#intervalBefore
	Fungsi	Mendeskripsikan <i>instance</i> sebagai <i>ProperInterval</i> yang terjadi sebelum <i>ProperInterval</i> lain.
<i>intervalAfter</i>	URI	http://www.w3.org/2006/time#intervalAfter
	Fungsi	Mendeskripsikan <i>instance</i> sebagai <i>ProperInterval</i> yang terjadi setelah <i>ProperInterval</i> lain.
<i>intervalDuring</i>	URI	http://www.w3.org/2006/time#intervalDuring
	Fungsi	Mendeskripsikan <i>instance</i> sebagai <i>ProperInterval</i> yang terjadi selama <i>ProperInterval</i> lain berlangsung.
<i>intervalContains</i>	URI	http://www.w3.org/2006/time#intervalContains
	Fungsi	Mendeskripsikan <i>instance</i> sebagai <i>ProperInterval</i> yang dimulai terlebih dahulu dan berakhir lebih lama dari <i>ProperInterval</i> lain.
<i>intervalStarts</i>	URI	http://www.w3.org/2006/time#intervalStarts

<i>Object Properties</i>		
	Fungsi	Mendekripsikan <i>instance</i> sebagai <i>ProperInterval</i> yang mengawali berlangsungnya <i>ProperInterval</i> lain.
<i>intervalStartedBy</i>	URI	http://www.w3.org/2006/time#intervalStartedBy
	Fungsi	Mendekripsikan <i>instance</i> sebagai <i>ProperInterval</i> yang diawali oleh <i>ProperInterval</i> lain.
<i>intervalFinishes</i>	URI	http://www.w3.org/2006/time#intervalFinishes
	Fungsi	Mendekripsikan <i>instance</i> sebagai <i>ProperInterval</i> yang mengakhiri berlangsungnya <i>ProperInterval</i> lain.
<i>intervalFinishedBy</i>	URI	http://www.w3.org/2006/time#intervalFinishedBy
	Fungsi	Mendekripsikan <i>instance</i> sebagai <i>ProperInterval</i> yang diakhiri oleh <i>ProperInterval</i> lain.
<i>intervalOverlaps</i>	URI	http://www.w3.org/2006/time#intervalOverlaps
	Fungsi	Mendekripsikan <i>instance</i> sebagai <i>ProperInterval</i> yang meng-overlap <i>ProperInterval</i> lain.
<i>intervalOverlappedBy</i>	URI	http://www.w3.org/2006/time#intervalOverlappedBy
	Fungsi	Mendekripsikan <i>instance</i> sebagai <i>ProperInterval</i> yang di-overlap oleh <i>ProperInterval</i> lain.
<i>intervalEquals</i>	URI	http://www.w3.org/2006/time#intervalEquals
	Fungsi	Mendekripsikan <i>instance</i> sebagai <i>ProperInterval</i> yang terjadi bersamaan dengan <i>ProperInterval</i> lain.
<i>overlaps</i>	URI	http://www.intelligence.tuc.gr/2011/timeEvents#overlaps

<i>Object Properties</i>		
	Fungsi	Mendekripsikan <i>instance</i> sebagai <i>ProperInterval</i> yang bersinggungan dengan <i>ProperInterval</i> lain.
<i>hasBeginning</i>	URI	http://www.w3.org/2006/time#hasBeginning
	Fungsi	Mendekripsikan <i>instance</i> sebagai <i>Instant</i> yang mengawali <i>ProperInterval</i>
<i>hasEnd</i>	URI	http://www.w3.org/2006/time#hasEnd
	Fungsi	Mendekripsikan <i>instance</i> sebagai <i>Instant</i> yang mengakhiri <i>ProperInterval</i>
<i>unitType</i>	URI	http://www.w3.org/2006/time#unitType
	Fungsi	Mendekripsikan tipe unit dari suatu <i>Instant</i>
<i>P4i_is_time-span_of</i>	URI	http://www.cidoc-crm.org/cidoc-crm/P4i_is_time-span_of
	Fungsi	Mendekripsikan <i>instance</i> sebagai <i>Time-Span</i> atau <i>TemporalEntity</i> suatu <i>Event</i> , <i>Thing</i> atau <i>Person</i> .
<i>P11i_participated_in</i>	URI	http://www.cidoc-crm.org/cidoc-crm/P11i_participated_in
	Fungsi	Mendesripsikan <i>instance</i> sebagai <i>Actor</i> yang berpartisipasi dalam <i>TemporalEntity</i>

Tabel 3.9 Penggunaan *Data Property* dari OWL-Time dan CIDOC-CRM

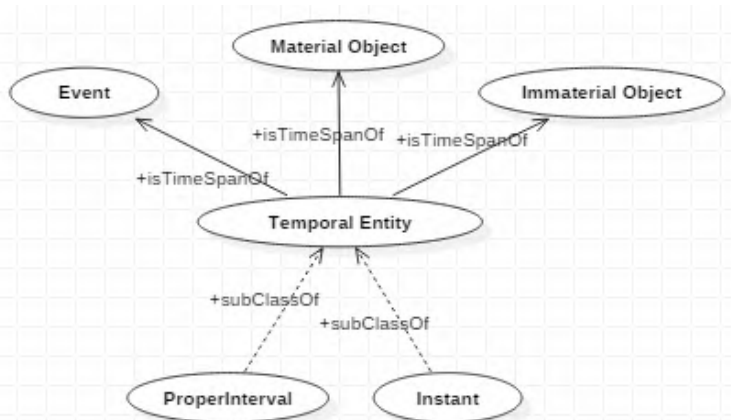
<i>Data Properties</i>		
<i>inXSDDateTime</i>	URI	http://www.w3.org/2006/time#inXSDDateTime
	Fungsi	Mendeskripsikan waktu dalam format <i>datetime</i> .

3.3.3 Penggabungan Ontologi

Entitas temporal pada ontologi hendaknya dihubungkan dengan kejadian (*Event*), objek material, maupun objek non-material yang terkait. Pada skema CIDOC-CRM, sebenarnya telah disediakan kelas *temporal entity* untuk menyimpan keterangan waktu tentang suatu fenomena, event atau kejadian. Akan tetapi, seperti yang sudah dijelaskan pada tahap pemilihan skema ontologi, konsep *temporal entity* milik OWL-Time lebih fleksibel dibandingkan dengan skema milik CIDOC-CRM.

Untuk mencari hubungan *event*, *person*, dan *thing* milik CIDOC-CRM dengan *temporal entity* milik OWL-Time, terlebih dahulu dilakukan pencarian properti yang terkait dengan *temporal entity* milik CIDOC-CRM. Pencarian ini dapat dilakukan dengan memanfaatkan fasilitas “*Class Usage*” yang merupakan bawaan dari Protégé 4.2. Pencarian dilakukan dengan mencari properti yang memiliki *domain* atau *range* dari kelas *event* dan *persistent item*. Dari pencarian yang dilakukan tidak ditemukan properti yang memiliki *domain* tersebut. Akan tetapi, terdapat dua properti yang *domain*-nya tidak didefinisikan yaitu ‘has time-span’ dan ‘is time-span of’. Itu berarti properti tersebut dapat diisi kelas *range* apa saja. Dari kedua properti tersebut yang paling tepat untuk menghubungkan entitas temporal dengan *event* dan *persistent item* adalah ‘is time-span of’. Dari penjelasan tersebut, dihasilkan bentuk umum penggabungan skema ontologi dari

OWL-Time dan CIDOC-CRM. Bentuk umum skema ontologi ditunjukkan oleh Gambar 3.9.



Gambar 3.9 Skema Inti Ontologi Hasil Penggabungan

3.3.4 *Equivalent Class* dan *Equivalent Property*

Kedua ontologi yang akan digabungkan memiliki beberapa kemiripan fungsi kelas seperti yang ditunjukkan oleh Tabel 3.10 dan kemiripan fungsi *object property* dan *data property* seperti yang ditunjukkan oleh Tabel 3.11.

Tabel 3.10 Daftar *Equivalent Classes*

Kelas CIDOC-CRM	Kelas OWL-Time
<i>E2_Temporal_Entity</i>	<i>TemporalEntity</i>
<i>E4_Period</i>	<i>Interval</i>
<i>E3_Condition_State</i>	<i>Instant</i>

Tabel 3.11 Daftar *Equivalent Properties*

CIDOC CRM Property	OWL-Time Property
<i>P114F.is_equal_in_time_to</i>	<i>intervalEquals</i>
<i>P115F.finishes</i>	<i>intervalFinishes</i>
<i>P115B.is_finished_by</i>	<i>intervalFinishedBy</i>

<i>P116F.starts</i>	<i>intervalStarts</i>
<i>P116B.is_started_by</i>	<i>intervalStartedBy</i>
<i>P117F.occurs_during</i>	<i>intervalDuring</i>
<i>P117B.includes</i>	<i>intervalContains</i>
<i>P118F.overlaps_in_time_with</i>	<i>intervalOverlaps</i>
<i>P118B.is_overlapped_in_time_by</i>	<i>intervalOverlappedBy</i>
<i>P119F.meets_in_time_with</i>	<i>intervalMeets</i>
<i>P119B.is_met_in_time_by</i>	<i>intervalMetBy</i>
<i>P120F.occurs_before</i>	<i>intervalBefore</i>
<i>P120B.occurs_after</i>	<i>intervalAfter</i>

Satu individual dapat didefinisikan sebagai dua kelas yang bersesuaian dengan menambahkan deskripsi “*Equivalent To*” melalui fasilitas yang disediakan Protégé. Dengan demikian, dengan mendefinisikan suatu individual sebagai satu *member* kelas A yang mana kelas A identik dengan kelas B, secara otomatis individual tersebut juga didefinisikan sebagai *member* kelas B setelah dilakukan proses *reasoning*. Proses tersebut berlaku juga untuk *object property* dan *data property*. Contoh penambahan deskripsi “*Equivalent To*” ditunjukkan oleh Gambar 3.10.



Gambar 3.10 Penambahan deskripsi “*Equivalent To*” pada Protégé

3.3.5 Pengembangan Ontologi

Hasil penggabungan kedua ontologi masih memiliki beberapa kekurangan. Suatu *TemporalEntity* dapat berupa suatu anotasi waktu dari suatu kejadian, atau merupakan suatu satuan waktu seperti tahun dan abad. Seperti yang sudah diketahui, data dalam studi kasus ini menyertakan instance abad dan tahun. Oleh sebab itu perlu dibedakan antara *TemporalEntity* yang merupakan abad, tahun dan *TemporalEntity* selain abad dan tahun. Kelas *Century* dan *Year* ditambahkan sebagai pembeda antara ketiganya. Kelas yang ditambahkan ditunjukkan oleh Tabel 3.12.

Tabel 3.12 Penambahan Kelas

Classes		
<i>Century</i>	URI	http://www.istory.id#Century
	Fungsi	Mendekripsikan <i>instance</i> sebagai abad
<i>Year</i>	URI	http://www.istory.id#Year
	Fungsi	Mendekripsikan <i>instance</i> sebagai tahun

3.4 Metode Pencarian Relasi

Pada sub bab ini dilakukan perancangan SWRL *rules* untuk pengecekan hubungan waktu antar entitas temporal dengan cara membandingkan *entity* satu dengan *entity* yang lain. Pengecekan dilakukan pada masukan data pada file ontologi dengan menggunakan bantuan SWRL *rule* dan bantuan Pellet *Reasoner*. SWRL *rule* digunakan untuk mendefinisikan fungsi aturan-aturan (*rules*) untuk mendapatkan fakta-fakta baru terkait hubungan waktu antar entitas temporal. Aturan-aturan dibuat berdasarkan *classes*, *object property*, dan *data property* pada skema ontologi yang dipakai dalam tugas akhir ini.

Dalam tugas akhir ini, pencarian hubungan antar entitas temporal dilakukan untuk mengetahui apakah sebuah entitas temporal memiliki hubungan dalam hal ini keterkaitan waktu dengan entitas temporal yang lain. Selain untuk mengetahui

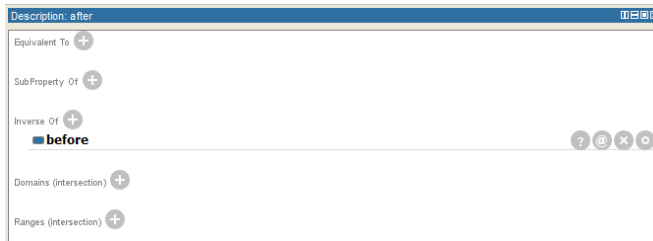
adanya keterkaitan waktu, jenis hubungan juga dapat diketahui melalui tahap ini. Jenis hubungan yang dimaksud adalah *before*, *after*, *equal*, *intervalBefore*, *intervalAfter*, *intervalMeets*, *intervalMetBy*, *intervalOverlaps*, *intervalOverlappedBy*, *intervalStarts*, *intervalStartedBy*, *intervalDuring*, *intervalContains*, *intervalFinishes*, *intervalFinishedBy*, *intervalEquals*.

Properti dalam ontologi ini memiliki relasi terhadap properti yang lain. Relasi-relasi tersebut ditambahkan dengan memanfaatkan *Characteristics* dan *Description*. Tabel 3.13 menunjukkan properti-properti yang memanfaatkan *Description* untuk melakukan pencarian relasi antar *temporal entity*.

Tabel 3.13 Relasi Properti dengan Menggunakan *Description*

<i>Property</i>	<i>Description</i>	<i>Value</i>
<i>before</i>	<i>Inverse Of</i>	<i>after</i>
<i>after</i>	<i>Inverse Of</i>	<i>before</i>
<i>intervalMeets</i>	<i>Inverse Of</i>	<i>intervalMetBy</i>
<i>intervalMetBy</i>	<i>Inverse Of</i>	<i>intervalMeets</i>
<i>intervalBefore</i>	<i>Inverse Of</i>	<i>intervalAfter</i>
<i>intervalAfter</i>	<i>Inverse Of</i>	<i>intervalBefore</i>
<i>intervalDuring</i>	<i>Inverse Of</i>	<i>intervalContains</i>
<i>intervalContains</i>	<i>Inverse Of</i>	<i>intervalDuring</i>
<i>intervalStarts</i>	<i>Inverse Of</i>	<i>intervalStartedBy</i>
<i>intervalStartedBy</i>	<i>Inverse Of</i>	<i>intervalStarts</i>
<i>intervalFinishes</i>	<i>Inverse Of</i>	<i>intervalFinishedBy</i>
<i>intervalFinishedBy</i>	<i>Inverse Of</i>	<i>intervalFinishes</i>
<i>intervalOverlaps</i>	<i>Inverse Of</i>	<i>intervalOverlappedBy</i>
<i>intervalOverlappedBy</i>	<i>Inverse Of</i>	<i>intervalOverlaps</i>

Penambahan *Description* pada properti dilakukan pada panel *Description* yang terdapat pada Protégé. Gambar 3.11 menunjukkan contoh penambahan *Description* pada *object property after* melalui Protégé.



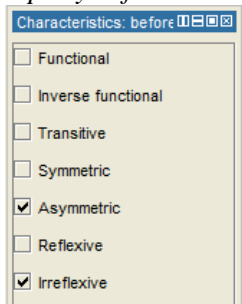
Gambar 3.11 Penambahan *Description* pada Protégé

Tabel Tabel 3.14 menunjukkan properti-properti yang memanfaatkan *Characteristic* untuk melakukan pencarian relasi antar *TemporalEntity*.

Tabel 3.14 Relasi Properti dengan Menggunakan *Characteristics*

<i>Property</i>	<i>Characteristic</i>
before	Asymmetric, Irreflexive
equals	Symmetric
IntervalEquals	Symmetric

Penambahan *Characteristic* pada properti dilakukan pada panel *Characteristic* yang terdapat pada Protégé. Gambar 3.12 menunjukkan contoh penambahan *Characteristic* pada *object property before* melalui Protégé.



Gambar 3.12 Penambahan *Characteristics* pada Protégé

3.4.1 Pencarian Relasi *before*

Relasi *before* dicari untuk mengetahui apakah suatu *Instant* berlangsung sebelum *Instant* yang lain.

3.4.1.1 Pencarian Dasar Relasi *before*

Rule 1 merupakan langkah dasar pencarian relasi *before*. Teknik pencarian dilakukan dengan membandingkan informasi waktu yang disimpan dalam masing-masing *Instant* sebagai *data property inXSDDateTime*. Informasi waktu pada individu kelas *Instant* disimpan dalam format *xsd:dateTime*. Pembandingan *xsd:dateTime* dilakukan dengan menggunakan fungsi *lessThan*.

Rule 1:

Instant(?x), Instant(?z), inXSDDateTime(?x, ?y), inXSDDateTime(?z, ?w),
lessThan(?y, ?w) -> before(?x, ?z)

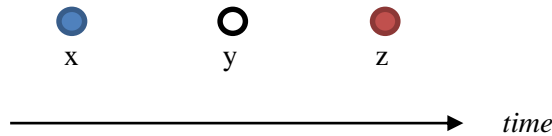
3.4.1.2 Pencarian Relasi *before* Kondisi 1

Rule 2 merupakan langkah pencarian relasi *before* dengan kondisi 1 yaitu dengan memanfaatkan informasi relasi *before* lain yang bersesuaian.

Rule 2:

before(?x, ?y), before(?y, ?z) -> before(?x, ?z)

Gambar 3.13 menunjukkan ilustrasi pencarian relasi *before* dengan kondisi 1 sesuai *Rule 2*. *Instant* x (warna biru) terjadi sebelum (*before*) *Instant* z (warna merah) ketika diketahui *Instant* x terjadi sebelum (*before*) *Instant* y (warna putih) dan *Instant* y terjadi sebelum (*before*) *Instant* z.



Gambar 3.13 Ilustrasi Relasi *before* Kondisi 1

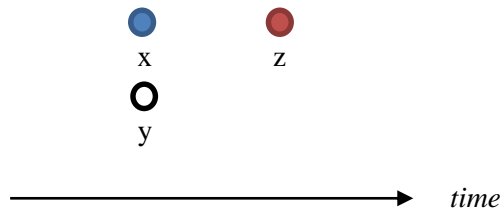
3.4.1.3 Pencarian Relasi *before* Kondisi 2

Rule 3 merupakan langkah pencarian relasi *before* dengan kondisi 2 yaitu dengan memanfaatkan informasi *equals* yang bersesuaian.

Rule 3:

$\text{equals}(\text{?x}, \text{?y}), \text{before}(\text{?y}, \text{?z}) \rightarrow \text{before}(\text{?x}, \text{?z})$

Gambar 3.14 menunjukkan ilustrasi pencarian relasi *before* dengan kondisi 2 sesuai *Rule 3*. *Instant x* (warna biru) terjadi sebelum (*before*) *Instant z* (warna merah) ketika diketahui *Instant x* terjadi bersamaan (*equals*) dengan *Instant y* (warna putih) dan *Instant y* terjadi sebelum (*before*) *Instant z*.



Gambar 3.14 Ilustrasi Relasi *before* Kondisi 2

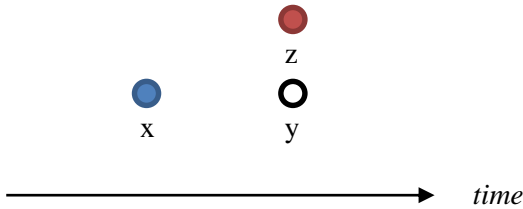
3.4.1.4 Pencarian Relasi *before* Kondisi 3

Rule 4 merupakan langkah pencarian relasi *before* dengan kondisi 3 yaitu dengan memanfaatkan informasi *equals* yang bersesuaian, akan tetapi dengan kondisi yang berbeda dengan *Rule 3*.

Rule 4:

$$\text{equals}(\text{?y}, \text{?z}), \text{before}(\text{?x}, \text{?y}) \rightarrow \text{before}(\text{?x}, \text{?z})$$

Gambar 3.15 menunjukkan ilustrasi pencarian relasi *before* dengan kondisi 3 sesuai *Rule 4*. *Instant x* (warna biru) terjadi sebelum (*before*) *Instant z* (warna merah) ketika diketahui *Instant y* (warna putih) terjadi bersamaan (*equals*) dengan *Instant z* dan *Instant x* (warna biru) terjadi sebelum (*before*) *Instant y*.



Gambar 3.15 Ilustrasi Relasi *before* Kondisi 3

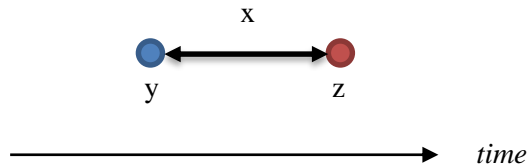
3.4.1.5 Pencarian Relasi *before* Kondisi 4

Rule 5 merupakan langkah pencarian relasi *before* dengan kondisi 4 yaitu ketika kedua titik terdapat dalam satu interval yang sama. Dalam satu *ProperInterval*, titik awal selalu terjadi lebih awal daripada titik akhirnya.

Rule 5:

$$\text{ProperInterval}(\text{?x}), \text{hasBeginning}(\text{?x}, \text{?y}), \text{hasEnd}(\text{?x}, \text{?z}) \rightarrow \text{before}(\text{?y}, \text{?z})$$

Gambar 3.16 menunjukkan gambaran pencarian relasi *before* dengan kondisi 4 sesuai *Rule 5*. *Instant y* (warna putih) terjadi sebelum (*before*) *Instant z* (warna merah) ketika *Instant y* (warna biru) adalah titik awal *ProperInterval x* dan *Instant z* adalah titik akhir *ProperInterval x*.



Gambar 3.16 Ilustrasi Relasi *before* Kondisi 4

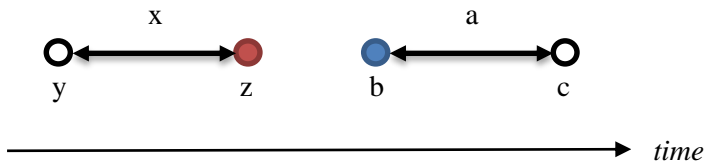
3.4.1.6 Pencarian Relasi *before* Kondisi 5

Rule 6 merupakan langkah pencarian relasi *before* dengan kondisi 5 yaitu ketika diketahui relasi *intervalBefore* yang bersesuaian. Titik akhir *ProperInterval* (?x) yang berlangsung lebih awal dari *ProperInterval* (?a) selalu terjadi lebih awal (*before*) dibanding titik akhir *ProperInterval* (?a).

Rule 6:

hasBeginning(?a, ?b), hasBeginning(?x, ?y), hasEnd(?a, ?c),
hasEnd(?x, ?z), intervalBefore(?x, ?a) -> before(?z, ?b)

Gambar 3.17 menunjukkan ilustrasi pencarian relasi *before* dengan kondisi 5 sesuai *Rule 6*. *Instant z* (warna merah) merupakan titik akhir *ProperInterval* x. *Instant b* (warna biru) merupakan titik awal *ProperInterval* a. *Instant z* terjadi sebelum (*before*) *Instant b* ketika *ProperInterval* x terjadi sebelum *ProperInterval* a.



Gambar 3.17 Ilustrasi Relasi *before* Kondisi 5

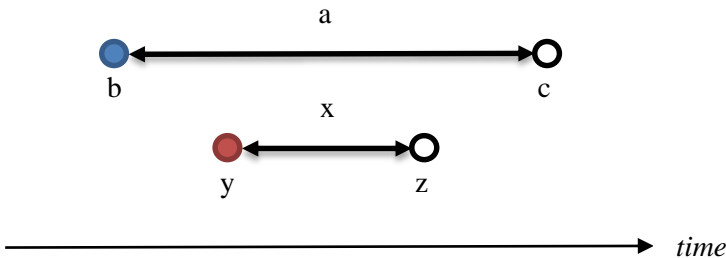
3.4.1.7 Pencarian Relasi *before* Kondisi 6

Rule 7 merupakan langkah pencarian relasi *before* dengan kondisi 6 yaitu ketika diketahui relasi *intervalDuring* yang bersesuaian. Titik awal suatu interval (?a) selalu lebih awal (*before*) dari titik awal interval (?x) yang terjadi di dalam rentang waktu interval (?a).

Rule 7:

hasBeginning(?a, ?b), hasBeginning(?x, ?y), hasEnd(?a, ?c),
hasEnd(?x, ?z), intervalDuring(?x, ?a) -> before(?b, ?y)

Gambar 3.18 menunjukkan gambaran pencarian relasi *before* dengan kondisi 6 sesuai *Rule 7*. *Instant* b (warna biru) merupakan titik awal *ProperInterval* a. *Instant* y (warna merah) merupakan titik awal *ProperInterval* x. *Instant* b terjadi sebelum (*before*) *Instant* y ketika *ProperInterval* x terjadi dalam rentang waktu *ProperInterval* a.



Gambar 3.18 Ilustrasi Relasi *before* Kondisi 6

3.4.1.8 Pencarian Relasi *before* Kondisi 7

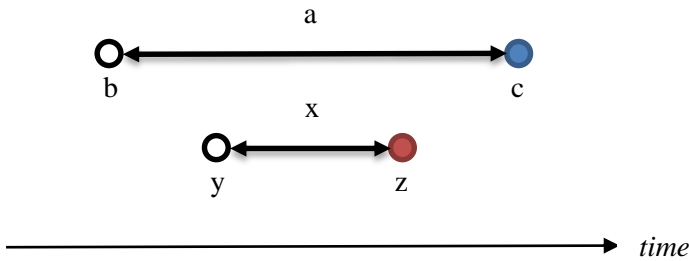
Rule 8 merupakan langkah pencarian relasi *before* dengan kondisi 7 yaitu ketika diketahui relasi *intervalDuring* yang bersesuaian pada titik yang berlainan dengan yang telah didefinisikan oleh *Rule 7*. Titik akhir suatu interval (?x) selalu

lebih awal (*before*) dari titik akhir interval (?a) yang terjadi di dalam rentang waktu interval (?x).

Rule 8:

hasBeginning(?a, ?b), hasBeginning(?x, ?y), hasEnd(?a, ?c),
hasEnd(?x, ?z), intervalDuring(?x, ?a) \rightarrow before(?z, ?c)

Gambar 3.19 menunjukkan gambaran pencarian relasi *before* dengan kondisi 7 sesuai *Rule 8*. *Instant c* (warna biru) merupakan titik akhir *ProperInterval a*. *Instant z* (warna merah) merupakan titik akhir *ProperInterval x*. *Instant z* terjadi sebelum (*before*) *Instant c* ketika *ProperInterval x* terjadi dalam rentang waktu *ProperInterval a*.



Gambar 3.19 Ilustrasi Relasi *before* Kondisi 7

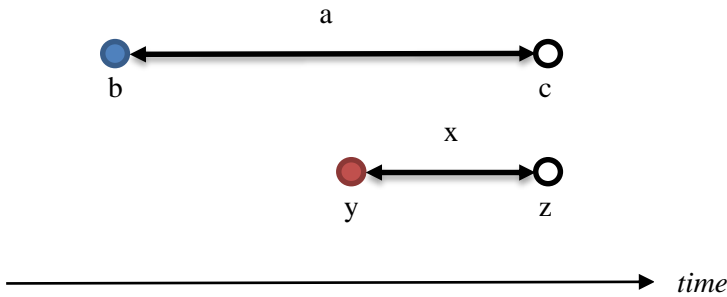
3.4.1.9 Pencarian Relasi *before* Kondisi 8

Rule 9 merupakan langkah pencarian relasi *before* dengan kondisi 8 yaitu ketika diketahui relasi *intervalFinishes* yang bersesuaian. Titik awal suatu interval (?a) selalu lebih awal (*before*) dari titik awal interval (?x) yang terjadi sebagai akhir rentang waktu interval (?a).

Rule 9:

hasBeginning(?a, ?b), hasBeginning(?x, ?y), hasEnd(?a, ?c),
hasEnd(?x, ?z), intervalFinishes(?x, ?a) \rightarrow before(?b, ?y)

Gambar 3.20 menunjukkan ilustrasi pencarian relasi *before* dengan kondisi 8 sesuai *Rule 9*. *Instant b* (warna biru) merupakan titik awal *ProperInterval a*. *Instant y* (warna merah) merupakan titik awal *ProperInterval x*. *Instant b* terjadi sebelum (*before*) *Instant y* ketika *ProperInterval x* mengakhiri rentang waktu *ProperInterval a*.



Gambar 3.20 Ilustrasi Relasi *before* Kondisi 8

3.4.1.10 Pencarian Relasi *before* Kondisi 9

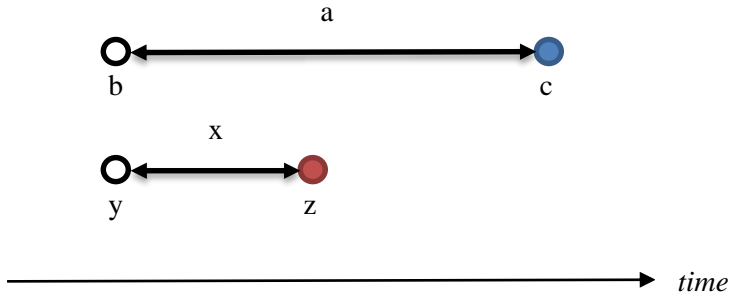
Rule 10 merupakan langkah pencarian relasi *before* dengan kondisi 9 yaitu ketika diketahui relasi *intervalStarts* yang bersesuaian. Titik akhir suatu interval (?x) yang terjadi sebagai awal rentang waktu interval (?a) selalu lebih awal (*before*) dari titik akhir interval (?a).

Rule 10:

hasBeginning(?a, ?b), hasBeginning(?x, ?y), hasEnd(?a, ?c),
hasEnd(?x, ?z), intervalStarts(?x, ?a) -> before(?z, ?c)

Gambar 3.21 menunjukkan ilustrasi pencarian relasi *before* dengan kondisi 9 sesuai *Rule 10*. *Instant c* (warna biru) merupakan titik akhir *ProperInterval a*. *Instant z* (warna merah) merupakan titik akhir *ProperInterval x*. *Instant z* terjadi

sebelum (*before*) *Instant c* ketika *ProperInterval x* mengakhiri rentang waktu *ProperInterval a*.



Gambar 3.21 Ilustrasi Relasi *before* Kondisi 9

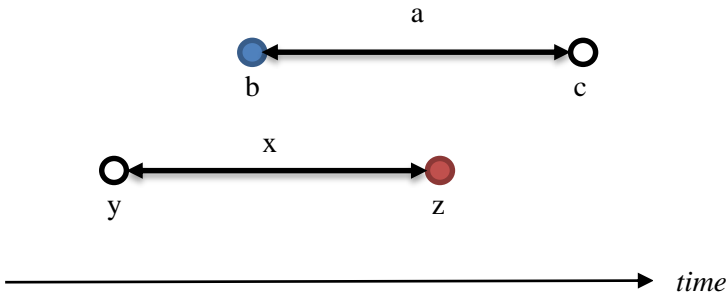
3.4.1.11 Pencarian Relasi *before* Kondisi 10

Rule 11 merupakan langkah pencarian relasi *before* dengan kondisi 10 yaitu ketika diketahui relasi *intervalOverlaps* yang bersesuaian. Titik awal sebuah interval (?a) selalu lebih awal (*before*) dari titik akhir interval (?x) yang meng-*overlap* interval (?a).

Rule 11:

hasBeginning(?a, ?b), hasBeginning(?x, ?y), hasEnd(?a, ?c),
hasEnd(?x, ?z), intervalOverlaps(?x, ?a) -> before(?b, ?z)

Gambar 3.22 menunjukkan ilustrasi pencarian relasi *before* dengan kondisi 10 sesuai *Rule 11*. *Instant b* (warna biru) merupakan titik awal *ProperInterval a*. *Instant z* (warna merah) merupakan titik akhir *ProperInterval x*. *Instant b* terjadi sebelum (*before*) *Instant z* ketika *ProperInterval x* meng-*overlap* *ProperInterval a*.



Gambar 3.22 Ilustrasi Relasi *before* Kondisi 10

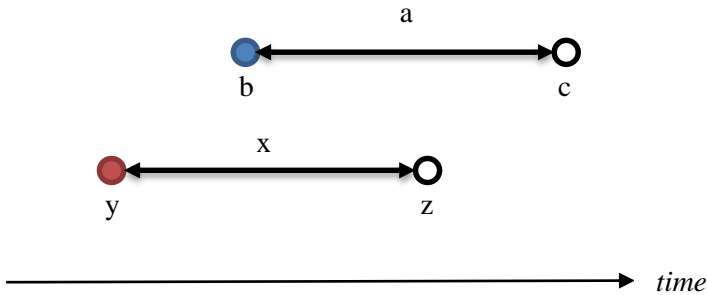
3.4.1.12 Pencarian Relasi *before* Kondisi 11

Rule 12 merupakan langkah pencarian relasi *before* dengan kondisi 11 yaitu ketika diketahui relasi *intervalOverlaps* yang bersesuaian. Titik akhir sebuah interval (*?x*) yang *overlap* interval (*?a*) selalu lebih awal (*before*) dari titik awal interval (*?a*).

Rule 12:

hasBeginning(*?a*, *?b*), hasBeginning(*?x*, *?y*), hasEnd(*?a*, *?c*),
hasEnd(*?x*, *?z*), intervalOverlaps(*?x*, *?a*) -> before(*?y*, *?b*)

Gambar 3.23 menunjukkan ilustrasi pencarian relasi *before* dengan kondisi 11 sesuai *Rule 12*. *Instant b* (warna biru) merupakan titik awal *ProperInterval a*. *Instant y* (warna merah) merupakan titik awal *ProperInterval x*. *Instant y* terjadi sebelum (*before*) *Instant b* ketika *ProperInterval x* *overlap* *ProperInterval a*.



Gambar 3.23 Ilustrasi Relasi *before* Kondisi 11

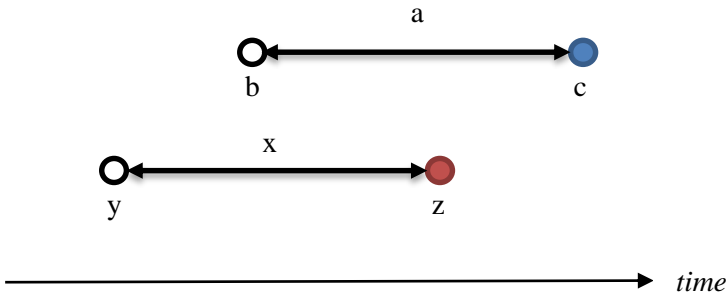
3.4.1.13 Pencarian Relasi *before* Kondisi 12

Rule 13 merupakan langkah pencarian relasi *before* dengan kondisi 12 yaitu ketika diketahui relasi *intervalOverlaps* yang bersesuaian. Titik akhir sebuah interval ($?x$) yang *overlap* interval ($?a$) selalu lebih awal (*before*) dari titik akhir interval ($?a$)

Rule 13:

hasBeginning($?a$, $?b$), hasBeginning($?x$, $?y$), hasEnd($?a$, $?c$),
hasEnd($?x$, $?z$), intervalOverlaps($?x$, $?a$) \rightarrow before($?z$, $?c$)

Gambar 3.24 menunjukkan ilustrasi pencarian relasi *before* dengan kondisi 12 sesuai *Rule 13*. *Instant c* (warna biru) merupakan titik akhir *ProperInterval a*. *Instant z* (warna merah) merupakan titik akhir *ProperInterval x*. *Instant z* terjadi sebelum (*before*) *Instant c* ketika *ProperInterval x* *overlap* *ProperInterval a*.

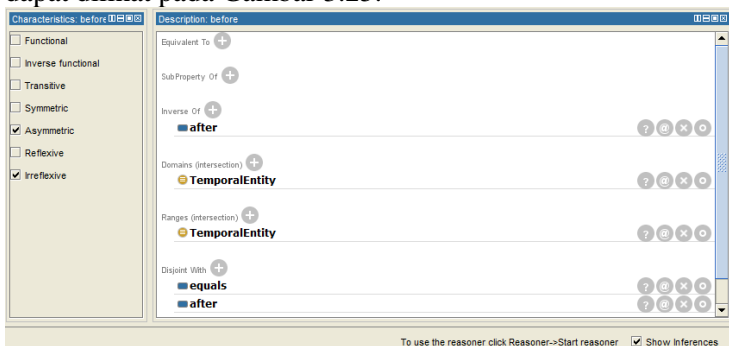


Gambar 3.24 Ilustrasi Relasi *before* Kondisi 12

3.4.2 Pencarian Relasi *after*

Relasi *after* dicari untuk mengetahui apakah suatu Instant berlangsung setelah *Instant* yang lain. Relasi *after* merupakan fungsi *invers* atau kebalikan dari relasi *before*. Suatu *Instant* X yang memiliki relasi *before* dengan range *Instant* Y sudah pasti *Instant* Y memiliki relasi *after* dengan range *Instant* X.

Invers dari suatu *triple* dapat diinisiasi dengan menambahkan fungsi *invers* pada *object property*, dalam hal ini *object property before*. Penambahan fungsi *invers* pada Protégé dapat dilihat pada Gambar 3.25.



Gambar 3.25 Penambahan Fungsi *Invers* pada Protégé

3.4.3 Pencarian Relasi *equals*

Relasi *equals* dicari untuk mengetahui apakah suatu *Instant* bernilai sama dengan *Instant* yang lain.

3.4.3.1 Dasar Pencarian Relasi *equals*

Rule 14 merupakan langkah dasar pencarian relasi *equals*. Teknik pencarian dilakukan dengan mengecek kesamaan informasi waktu yang disimpan dalam masing-masing *Instant* sebagai *data property inXSDDateTime*. Informasi waktu pada individu kelas *Instant* disimpan dalam format *xsd:dateTime*. Pengecekan kesamaan *xsd:dateTime* dilakukan dengan menggunakan fungsi *equal*.

Rule 14:

Instant(?x), Instant(?z), inXSDDateTime(?x, ?y), inXSDDateTime(?z, ?w),
equal(?y, ?w) -> equals(?z, ?x)

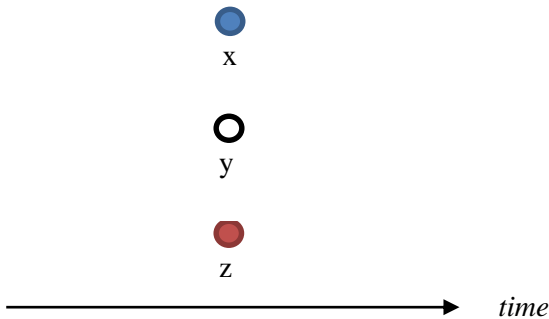
3.4.3.2 Pencarian Relasi *equals* Kondisi 1

Rule 15 merupakan langkah pencarian relasi *equals* dengan kondisi 1 yaitu dengan memanfaatkan informasi relasi *equals* lain yang bersesuaian.

Rule 15:

equals(?x, ?y), equals(?y, ?z) -> equals(?x, ?z)

Gambar 3.26 menunjukkan ilustrasi pencarian relasi *equals* dengan kondisi 1 sesuai *Rule 15*. *Instant x* (warna biru) terjadi bersamaan (*equals*) dengan *Instant z* (warna merah) ketika diketahui *Instant x* terjadi bersamaan (*equals*) dengan *Instant y* (warna putih) dan *Instant y* terjadi bersamaan (*equals*) dengan *Instant z*.



Gambar 3.26 Ilustrasi Relasi *equals* Kondisi 1

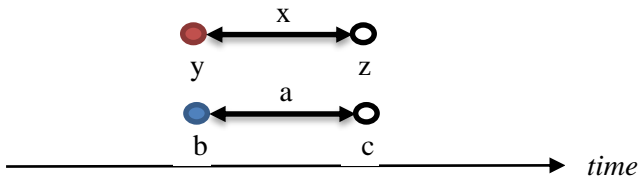
3.4.3.3 Pencarian Relasi *equals* Kondisi 2

Rule 16 merupakan langkah pencarian relasi *before* dengan kondisi 2 yaitu ketika diketahui relasi *intervalEquals* yang bersesuaian. Titik awal suatu interval (?a) selalu sama dengan titik awal interval (?x) yang terjadi bersamaan dengan interval (?a).

Rule 16:

hasBeginning(?a, ?b), hasBeginning(?x, ?y), hasEnd(?a, ?c),
hasEnd(?x, ?z), intervalEquals(?x, ?a) -> equals(?b, ?y)

Gambar 3.27 menunjukkan ilustrasi pencarian relasi *equals* dengan kondisi 2 sesuai *Rule 16*. *Instant b* (warna biru) merupakan titik awal *ProperInterval a*. *Instant y* (warna merah) merupakan titik awal *ProperInterval x*. *Instant b* terjadi bersamaan (*equals*) dengan *Instant y* ketika *ProperInterval x* terjadi bersamaan (*IntervalEquals*) dengan *ProperInterval a*.



Gambar 3.27 Ilustrasi Relasi *equals* Kondisi 2

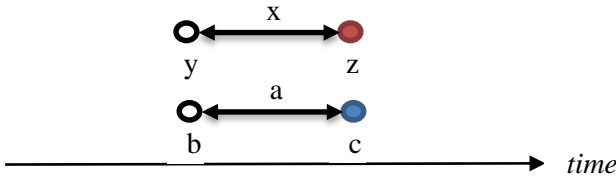
3.4.3.4 Pencarian Relasi *equals* Kondisi 3

Rule 17 merupakan langkah pencarian relasi *equals* dengan kondisi 3 yaitu ketika diketahui relasi *intervalEquals* yang bersesuaian. Titik akhir suatu interval (?a) selalu sama dengan titik akhir interval (?x) yang terjadi bersamaan dengan interval (?a).

Rule 17:

hasBeginning(?a, ?b), hasBeginning(?x, ?y), hasEnd(?a, ?c),
hasEnd(?x, ?z), intervalEquals(?x, ?a) -> equals(?c, ?z)

Gambar 3.28 menunjukkan ilustrasi pencarian relasi *equals* dengan kondisi 3 sesuai *Rule 17*. *Instant c* (warna biru) merupakan titik akhir *ProperInterval a*. *Instant z* (warna merah) merupakan titik akhir *ProperInterval x*. *Instant c* terjadi bersamaan (*equals*) dengan *Instant z* ketika *ProperInterval x* terjadi bersamaan dengan *ProperInterval a*.



Gambar 3.28 Ilustrasi Relasi *equals* Kondisi 3

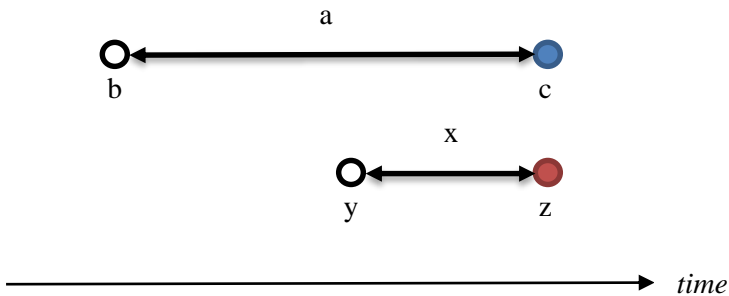
3.4.3.5 Pencarian Relasi *equals* Kondisi 4

Rule 18 merupakan langkah pencarian relasi *equals* dengan kondisi 4 yaitu ketika diketahui relasi *intervalFinishes* yang bersesuaian. Titik akhir suatu interval (?a) selalu sama dengan titik akhir interval (?x) yang mengakhiri rentang waktu interval (?a).

Rule 18:

hasBeginning(?a, ?b), hasBeginning(?x, ?y), hasEnd(?a, ?c),
hasEnd(?x, ?z), intervalFinishes(?x, ?a) -> equals(?z, ?c)

Gambar 3.29 menunjukkan ilustrasi pencarian relasi *equals* dengan kondisi 4 sesuai *Rule 18*. *Instant c* (warna biru) merupakan titik akhir *ProperInterval a*. *Instant z* (warna merah) merupakan titik akhir *ProperInterval x*. *Instant c* terjadi bersamaan (*equals*) dengan *Instant z* ketika *ProperInterval x* mengakhiri rentang waktu *ProperInterval a*.



Gambar 3.29 Ilustrasi Relasi *equals* Kondisi 4

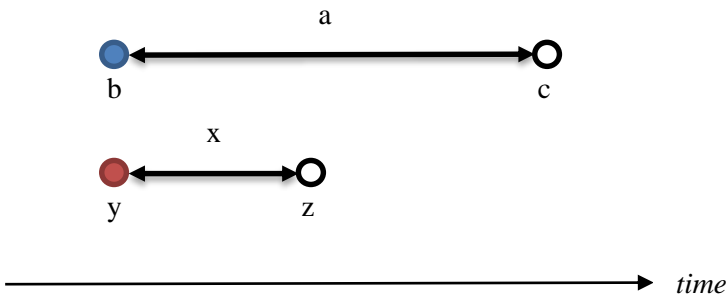
3.4.3.6 Pencarian Relasi *equals* Kondisi 5

Rule 19 merupakan langkah pencarian relasi *equals* dengan kondisi 5 yaitu ketika diketahui relasi *intervalStarts* yang bersesuaian. Titik awal suatu interval (?a) selalu sama dengan titik awal interval (?x) yang memulai rentang waktu interval (?a).

Rule 19:

hasBeginning(?a, ?b), hasBeginning(?x, ?y), hasEnd(?a, ?c),
hasEnd(?x, ?z), intervalStarts(?x, ?a) -> equals(?y, ?b)

Gambar 3.30 menunjukkan ilustrasi pencarian relasi *equals* dengan kondisi 5 sesuai *Rule 19*. *Instant b* (warna biru) merupakan titik awal *ProperInterval a*. *Instant y* (warna merah) merupakan titik awal *ProperInterval x*. *Instant b* terjadi bersamaan (*equals*) dengan *Instant y* ketika *ProperInterval x* mengakhiri rentang waktu *ProperInterval a*.



Gambar 3.30 Ilustrasi Relasi *equals* Kondisi 5

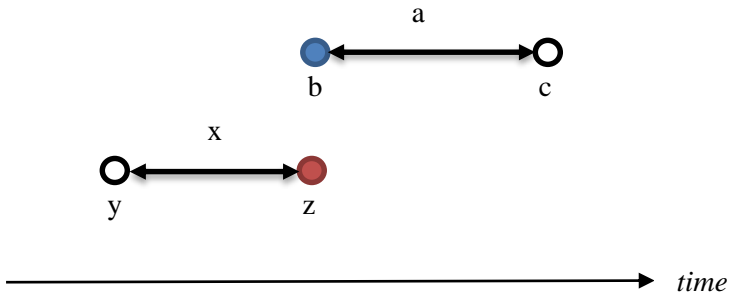
3.4.3.7 Pencarian Relasi *equals* Kondisi 6

Rule 20 merupakan langkah pencarian relasi *equals* dengan kondisi 6 yaitu ketika diketahui relasi *intervalMeets* yang bersesuaian. Titik awal suatu interval (*?a*) selalu sama dengan titik awal interval (*?x*) yang memulai rentang waktu interval (*?a*).

Rule 20:

$\text{hasBeginning}(\text{?a}, \text{?b}), \text{hasBeginning}(\text{?x}, \text{?y}), \text{hasEnd}(\text{?a}, \text{?c}), \text{hasEnd}(\text{?x}, \text{?z}), \text{intervalMeets}(\text{?x}, \text{?a}) \rightarrow \text{equals}(\text{?z}, \text{?b})$

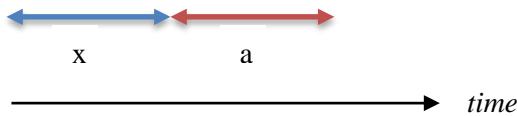
Gambar 3.31 menunjukkan ilustrasi pencarian relasi *equals* dengan kondisi 6 sesuai *Rule 20*. *Instant b* (warna biru) merupakan titik awal *ProperInterval a*. *Instant z* (warna merah) merupakan titik akhir *ProperInterval x*. *Instant b* terjadi bersamaan (*equals*) dengan *Instant z* ketika *ProperInterval x* bertemu dengan *ProperInterval a*.



Gambar 3.31 Ilustrasi Relasi *equals* Kondisi 6

3.4.4 Pencarian Relasi *intervalMeets*

Relasi *intervalMeets* dicari untuk mengetahui apakah suatu interval bertemu dengan interval yang lain. Dalam konteks ini, bertemu memiliki arti bahwa titik akhir suatu interval memiliki nilai yang sama dengan nilai awal interval lain. Gambar 3.32 menunjukkan ilustrasi dari relasi *intervalMeets*(?x, ?a).



Gambar 3.32 Ilustrasi Relasi *intervalMeets*

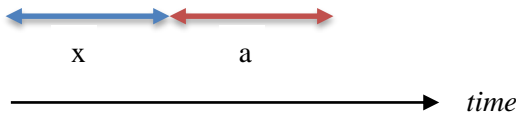
SWRL rule untuk pencarian *intervalMeets* dijelaskan dalam *Rule 21*.

Rule 21:

ProperInterval(?a), ProperInterval(?x), equals(?z, ?b),
hasBeginning(?a, ?b), hasEnd(?x, ?z) -> intervalMeets(?x, ?a)

3.4.5 Pencarian Relasi *intervalMetBy*

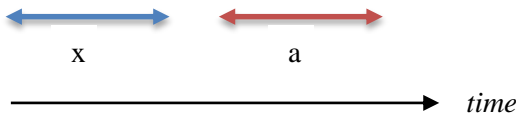
Relasi *intervalMetBy* dicari dengan menambahkan fungsi *invers* dengan nilai *intervalMeets*. Suatu interval X yang memiliki relasi *intervalMeets* dengan *range* interval Y sudah pasti interval Y memiliki relasi *intervalMetBy* dengan *range* interval X. Gambar 3.33 menunjukkan ilustrasi dari relasi *intervalBefore*(?a, ?x).



Gambar 3.33 Ilustrasi Relasi *intervalMetBy*

3.4.6 Pencarian Relasi *intervalBefore*

Relasi *intervalBefore* dicari untuk mengetahui apakah suatu interval terjadi sebelum interval yang lain. Gambar 3.34 menunjukkan ilustrasi dari relasi *intervalBefore*(?x, ?a).



Gambar 3.34 Ilustrasi Relasi *intervalBefore*

SWRL rule untuk pencarian *intervalBefore* dijelaskan dalam Rule 22.

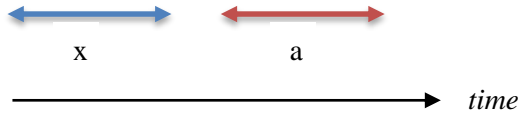
Rule 22:

ProperInterval(?a), ProperInterval(?x), before(?z, ?b),
hasBeginning(?a, ?b), hasEnd(?x, ?z) -> intervalBefore(?x, ?a)

3.4.7 Pencarian Relasi *intervalAfter*

Relasi *intervalAfter* dicari dengan menambahkan fungsi *invers* dengan nilai *intervalBefore*. Suatu interval X yang

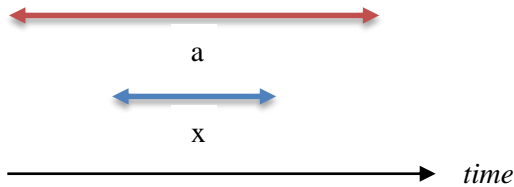
memiliki relasi *intervalBefore* dengan *range* interval Y sudah pasti interval Y memiliki relasi *intervalAfter* dengan *range* interval X. Gambar 3.35 menunjukkan ilustrasi dari relasi *intervalAfter*(?a, ?x).



Gambar 3.35 Ilustrasi Relasi *intervalAfter*

3.4.8 Pencarian Relasi *intervalDuring*

Relasi *intervalDuring* dicari untuk mengetahui apakah suatu interval terjadi dalam rentang waktu interval yang lain. Gambar 3.36 menunjukkan ilustrasi dari relasi *intervalDuring*(?x, ?a).



Gambar 3.36 Ilustrasi Relasi *intervalDuring*

SWRL rule untuk pencarian *intervalDuring* dijelaskan dalam Rule 23.

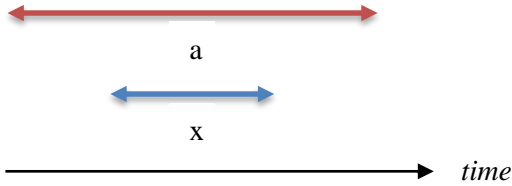
Rule 23:

ProperInterval(?a), ProperInterval(?x), before(?b, ?y), before(?z, ?c),
hasBeginning(?a, ?b), hasBeginning(?x, ?y), hasEnd(?a, ?c),
hasEnd(?x, ?z) -> intervalDuring(?x, ?a)

3.4.9 Pencarian Relasi *intervalContains*

Relasi *intervalContains* dicari dengan menambahkan fungsi *invers* dengan nilai *intervalDuring*. Suatu interval X

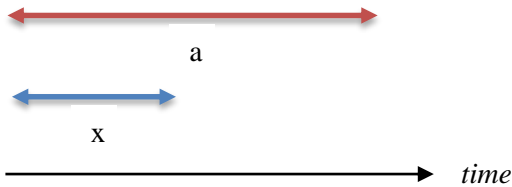
yang memiliki relasi *intervalDuring* dengan *range* interval Y sudah pasti interval Y memiliki relasi *intervalContains* dengan *range* interval X. Gambar 3.37 menunjukkan ilustrasi dari relasi *intervalContains*(?a, ?x).



Gambar 3.37 Ilustrasi Relasi *intervalContains*

3.4.10 Pencarian Relasi *intervalStarts*

Relasi *intervalStarts* dicari untuk mengetahui apakah suatu interval mengawali rentang waktu interval yang lain. Gambar 3.38 menunjukkan ilustrasi dari relasi *intervalStarts*(?x, ?a).



Gambar 3.38 Ilustrasi Relasi *intervalStarts*

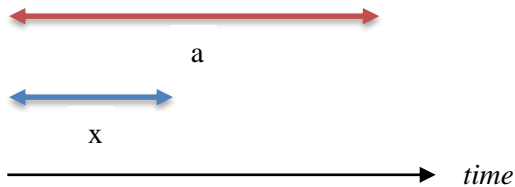
SWRL rule untuk pencarian *intervalStarts* dijelaskan dalam Rule 24.

Rule 24:

ProperInterval(?a), ProperInterval(?x), equals(?y, ?b), before(?z, ?c),
hasBeginning(?a, ?b), hasBeginning(?x, ?y), hasEnd(?a, ?c),
hasEnd(?x, ?z) -> intervalStarts(?x, ?a)

3.4.11 Pencarian Relasi *intervalStartedBy*

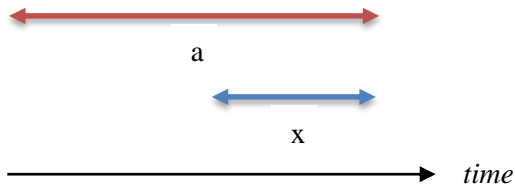
Relasi *intervalStartedBy* dicari dengan menambahkan fungsi *invers* dengan nilai *intervalStarts*. Suatu interval X yang memiliki relasi *intervalStarts* dengan *range* interval Y sudah pasti interval Y memiliki relasi *intervalStartedBy* dengan *range* interval X. Gambar 3.39 menunjukkan ilustrasi dari relasi *intervalStartedBy*(?a, ?x).



Gambar 3.39 Ilustrasi Relasi *intervalStartedBy*

3.4.12 Pencarian Relasi *intervalFinishes*

Relasi *intervalFinishes* dicari untuk mengetahui apakah suatu interval mengakhiri rentang waktu interval yang lain. Gambar 3.40 menunjukkan ilustrasi dari relasi *intervalFinishes*(?x, ?a).



Gambar 3.40 Ilustrasi Relasi *intervalFinishes*

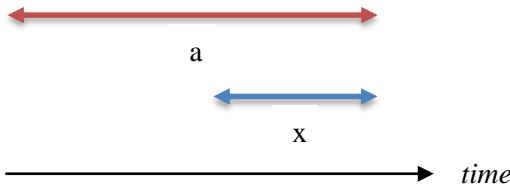
SWRL rule untuk pencarian *intervalFinishes* dijelaskan dalam *Rule 21*.

Rule 25:

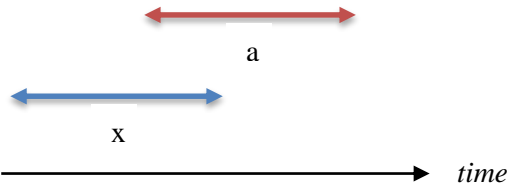
ProperInterval(?a), ProperInterval(?x), equals(?z, ?c), before(?b, ?y),
 hasBeginning(?a, ?b), hasBeginning(?x, ?y), hasEnd(?a, ?c),
 hasEnd(?x, ?z) -> intervalFinishes(?x, ?a)

3.4.13 Pencarian Relasi *intervalFinishedBy*

Relasi *intervalFinishedBy* dicari dengan menambahkan fungsi *invers* dengan nilai *intervalFinishes*. Suatu interval X yang memiliki relasi *intervalFinishes* dengan *range* interval Y sudah pasti interval Y memiliki relasi *intervalFinishedBy* dengan *range* interval X. Gambar 3.41 menunjukkan ilustrasi dari relasi *intervalFinishedBy*(?a, ?x).

**Gambar 3.41 Ilustrasi Relasi *intervalFinishedBy*****3.4.14 Pencarian Relasi *intervalOverlaps***

Relasi *intervalOverlaps* dicari untuk mengetahui apakah suatu interval meng-overlap interval yang lain. Gambar 3.42 menunjukkan ilustrasi dari relasi *intervalOverlaps*(?x, ?a).

**Gambar 3.42 Ilustrasi Relasi *intervalOverlaps***

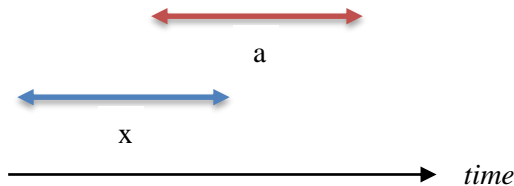
SWRL rule untuk pencarian *intervalOverlaps* dijelaskan dalam *Rule 26*.

Rule 26:

ProperInterval(?a), ProperInterval(?x), before(?b, ?z), before(?y, ?b), before(?z, ?c), hasBeginning(?a, ?b), hasBeginning(?x, ?y), hasEnd(?a, ?c), hasEnd(?x, ?z) -> intervalOverlaps(?x, ?a)

3.4.15 Pencarian Relasi *intervalOverlappedBy*

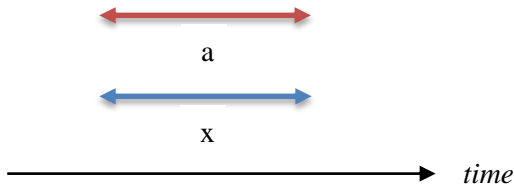
Relasi *intervalOverlappedBy* dicari dengan menambahkan fungsi *invers* dengan nilai *intervalOverlaps*. Suatu interval X yang memiliki relasi *intervalOverlaps* dengan *range* interval Y sudah pasti interval Y memiliki relasi *intervalOverlappedBy* dengan *range* interval X. Gambar 3.43 menunjukkan ilustrasi dari relasi *intervalOverlappedBy*(?a, ?x).



Gambar 3.43 Ilustrasi Relasi *intervalOverlappedBy*

3.4.16 Pencarian Relasi *intervalEquals*

Relasi *intervalEquals* dicari untuk mengetahui apakah suatu interval bersamaan dengan interval yang lain. Gambar 3.44 menunjukkan ilustrasi dari relasi *intervalEquals*(?x, ?a).



Gambar 3.44 Ilustrasi Relasi *intervalEquals*

SWRL rule untuk pencarian *intervalEquals* dijelaskan dalam *Rule 27*.

Rule 27:

ProperInterval(?a), ProperInterval(?x), equals(?y, ?b), equals(?z, ?c),
hasBeginning(?a, ?b), hasBeginning(?x, ?y), hasEnd(?a, ?c),
hasEnd(?x, ?z), DifferentFrom (?a, ?x) -> intervalEquals(?x, ?a)

BAB IV

ANALISIS DAN PERANCANGAN SISTEM

Pada bab ini dijelaskan tentang analisis permasalahan dan perancangan Tugas Akhir. Analisis permasalahan membahas tentang permasalahan yang diangkat dalam Tugas Akhir ini beserta solusi yang ditawarkan. Selanjutnya dibahas juga tentang perancangan sistem yang dibuat.

4.1 Analisis

Tahap analisis dibagi menjadi beberapa bagian antara lain cakupan permasalahan, deskripsi umum sistem, kasus penggunaan sistem dan kebutuhan perangkat lunak.

4.1.1 Cakupan Permasalahan

Permasalahan utama yang diangkat dalam tugas akhir ini adalah pencarian relasi antar objek-objek warisan budaya Indonesia. Studi kasus direpresentasikan dengan suatu skema model ontologi. Pencarian relasi semantik waktu dilakukan dengan menggunakan *SWRL rule*. Proses *reasoning* dilakukan untuk mendapatkan fakta-fakta baru yang kemudian fakta-fakta baru disimpan kembali sebagai ontologi baru. Keluaran dari proses tersebut berupa *file* RDF. Dibutuhkan sebuah sistem untuk membantu menampilkan hasil *reasoning* ontologi yang dirancang. Sistem yang dibuat harus dapat menampilkan hasil ontologi dengan tampilan yang mudah dipahami oleh pengguna.

4.1.2 Deskripsi Umum Sistem

Aplikasi iStory dibangun untuk memudahkan pembacaan file ontologi yang dibuat pada tugas akhir ini. Aplikasi iStory dibangun dengan basis *web* menggunakan bahasa pemrograman PHP dan library EasyRDF. Masukan dari perangkat lunak ini berupa file RDF yang merupakan hasil ekspor ontologi yang telah dilakukan *reasoning*. Keluaran dari

perangkat lunak ini adalah sebuah halaman HTML yang menampilkan informasi dari *file* RDF.

4.1.3 Spesifikasi Kebutuhan Perangkat Lunak

Bab ini menjelaskan kebutuhan perangkat lunak dalam bentuk diagram kasus dan diagram aktivitas. Masing-masing diagram menjelaskan perilaku atau sifat dari sistem ini.

4.1.3.1 Kebutuhan Fungsional

Kebutuhan fungsional adalah kebutuhan pokok yang harus dipenuhi agar sistem dapat berjalan dengan baik. Daftar kebutuhan fungsional dapat dilihat pada Tabel 4.1.

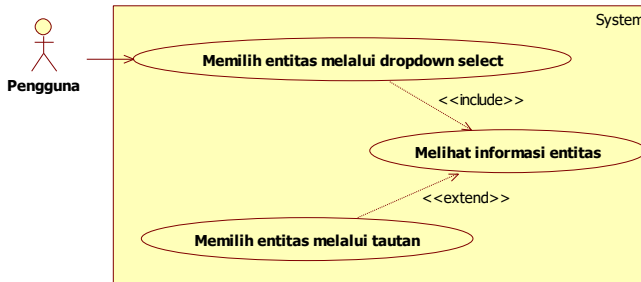
Tabel 4.1 Kebutuhan Fungsional Sistem

Kode Kebutuhan	Kebutuhan Fungsional	Deskripsi
TA-KF0001	Memilih entitas melalui <i>combo box</i>	Pengguna dapat memilih entitas yang ditampilkan melalui <i>combo box</i>
TA-KF0002	Memilih entitas melalui tautan	Pengguna dapat memilih entitas yang ditampilkan melalui tautan
TA-KF0003	Melihat informasi entitas	Pengguna dapat melihat informasi dari entitas yang dipilih

4.1.4 Aktor

Aktor adalah entitas-entitas yang terlibat dan berinteraksi langsung dengan sistem. Entitas ini bisa berupa manusia maupun sistem atau perangkat lunak yang lain. Aktor dalam Tugas Akhir ini yaitu pengguna yang diasumsikan tidak memahami bahasa pemrograman. Pengguna dapat melihat informasi semantik waktu dari entitas yang dipilih melalui *combo box* maupun tautan yang disediakan oleh sistem.

4.1.5 Kasus Penggunaan



Gambar 4.1 Diagram Kasus Penggunaan

Kasus penggunaan dijabarkan dalam bentuk spesifikasi kasus penggunaan dan diagram aktivitas. Diagram kasus penggunaan dapat dilihat pada Gambar 4.1. Pengguna dapat memilih entitas melalui *dropdown select*, kemudian melihat informasi dari entitas yang dipilih sehingga keduanya memiliki hubungan *include*. Pengguna juga bisa memilih entitas untuk ditampilkan informasinya melalui halaman deskripsi yang sudah ditampilkan melalui halaman deskripsi sehingga keduanya memiliki hubungan *extend*. Daftar kode diagram kasus penggunaan sistem dapat dilihat pada Tabel 4.2 Daftar Kode Diagram Kasus Penggunaan.

Tabel 4.2 Daftar Kode Diagram Kasus Penggunaan

Kode Kasus Penggunaan	Nama
TA-UC0001	Memilih entitas melalui <i>combo box</i>
TA-UC0002	Memilih entitas melalui tautan
TA-UC0003	Melihat informasi entitas

4.1.5.1 Memilih Entitas Melalui *Combo Box*

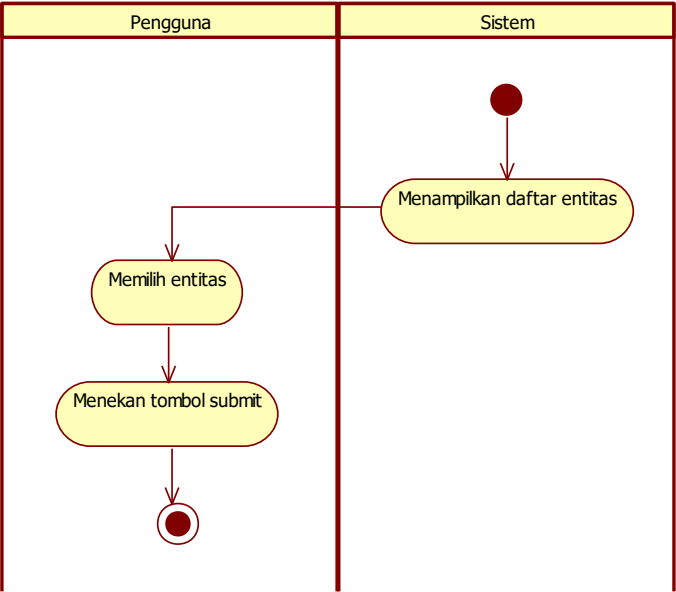
Pada kasus penggunaan ini, sistem akan menangkap daftar entitas dari file ontologi untuk kemudian ditampilkan kepada pengguna dalam bentuk *combo box* HTML agar dapat

dipilih oleh pengguna. Spesifikasi kasus penggunaannya dapat dilihat pada Tabel 4.3. Diagram aktivitasnya dapat dilihat pada Gambar 4.2.

Tabel 4.3 Spesifikasi Kasus Penggunaan Memilih Entitas Melalui *Combo Box*

Nama	Memilih entitas melalui <i>combo box</i>
Kode	TA-UC0001
Deskripsi	Menampilkan daftar entitas dari file ontologi dalam bentuk <i>combo box</i>
Tipe	Fungsional
Pemicu	Pengguna menekan tombol <i>submit</i> yang ditampilkan sistem
Aktor	Pengguna
Kondisi Awal	Sudah terdapat <i>file rdf</i> dalam sistem.
Aliran: - Kejadian Normal	<ol style="list-style-type: none"> 1. Sistem menampilkan daftar entitas 2. Pengguna memilih entitas melalui <i>combo box</i> 3. Pengguna menekan tombol <i>submit</i>
- Kejadian Alternatif	
Kondisi Akhir	Sistem menampilkan daftar entitas yang dipilih dalam bentuk tabel dan <i>list</i> .

Kebutuhan Khusus	Tidak ada
------------------	-----------



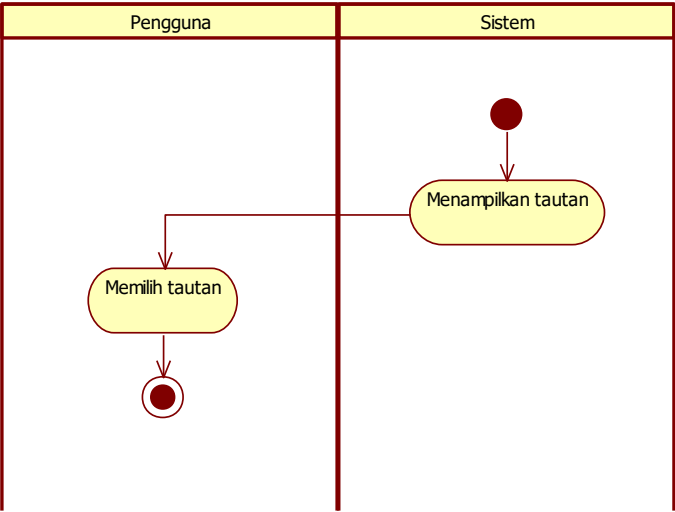
Gambar 4.2 Diagram Aktivitas Kasus Penggunaan Memilih Entitas Melalui *Combo Box*

4.1.5.2 Memilih Entitas Melalui Tautan

Pada kasus penggunaan ini, sistem akan menangkap daftar entitas dari file ontologi untuk kemudian ditampilkan kepada pengguna dalam bentuk tautan (*link*) agar dapat dipilih oleh pengguna. Spesifikasi kasus penggunaannya dapat dilihat pada Tabel 4.4. Diagram aktivitasnya dapat dilihat pada Gambar 4.3.

Tabel 4.4 Spesifikasi Kasus Penggunaan Memilih Entitas Melalui Tautan

Nama	Memilih entitas melalui tautan
Kode	TA-UC0002
Deskripsi	Menampilkan daftar entitas dari file ontologi dalam bentuk tautan
Tipe	Fungsional
Pemicu	Pengguna menekan tautan yang ditampilkan sistem
Aktor	Pengguna
Kondisi Awal	Sistem sudah menampilkan informasi entitas yang dipilih melalui <i>combo box</i>
Aliran: - Kejadian Normal	<ol style="list-style-type: none"> 1. Sistem menampilkan tautan entitas 2. Pengguna memilih tautan
- Kejadian Alternatif	
Kondisi Akhir	Sistem menampilkan daftar entitas yang dipilih dalam bentuk tabel dan <i>list</i> .
Kebutuhan Khusus	Tidak ada



Gambar 4.3 Diagram Aktivitas Kasus Penggunaan Memilih Entitas Melalui Tautan

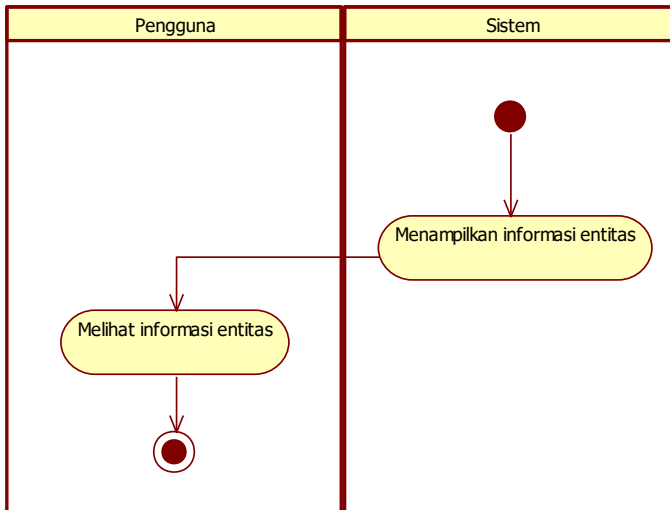
4.1.5.3 Melihat Informasi Entitas

Pada kasus penggunaan ini, sistem akan mengambil informasi mengenai entitas yang dipilih ke file ontologi untuk kemudian ditampilkan ke dalam sebuah halaman HTML dalam bentuk tabel dan *list*. Spesifikasi kasus penggunaannya dapat dilihat pada . Diagram aktivitasnya dapat dilihat pada.

Tabel 4.5 Spesifikasi Kasus Penggunaan Melihat Informasi Entitas

Nama	Melihat informasi entitas
Kode	TA-UC0003

Deskripsi	Menampilkan informasi semantik waktu dari entitas yang dipilih ke dalam tabel dan <i>list</i> HTML
Tipe	Fungsional
Pemicu	Pengguna memilih entitas melalui <i>combo box</i> atau tautan.
Aktor	Pengguna
Kondisi Awal	Sudah terdapat <i>file</i> rdf dalam sistem.
Aliran: - Kejadian Normal	1. Pengguna memilih entitas melalui <i>combo box</i> 2. Pengguna menekan tombol <i>submit</i>
- Kejadian Alternatif	
Kondisi Akhir	Sistem menampilkan informasi dari entitas yang dipilih dalam bentuk tabel dan <i>list</i>
Kebutuhan Khusus	Tidak ada



Gambar 4.4 Diagram Aktivitas Kasus Penggunaan Melihat Informasi Entitas

4.2 Perancangan Antarmuka Pengguna

Bagian ini membahas mengenai perancangan antarmuka yang akan dibuat. Rancangan antarmuka dibuat agar semudah mungkin dapat dipahami dan digunakan oleh pengguna.

Antarmuka aplikasi iStory terdiri dari satu halaman yang terdiri dari satu buah panel berisi *dropdown select* dan satu panel berisi deskripsi entitas. Deskripsi entitas sendiri terdiri dari beberapa tabel yang memuat informasi mengenai entitas yang dipilih. Rancangan antarmuka halaman utama ini dapat dilihat pada Gambar 4.5 dan Gambar 4.6. Penjelasan mengenai atribut-atribut yang terdapat pada halaman ini bisa dilihat pada Tabel 4.6

1

	>
Item 1	>
Item 2	>
Item 3	>
Item 4	>

2

Submit

Gambar 4.5 Rancangan Antarmuka Halaman Utama 1

3

4

Submit

Gambar 4.6 Rancangan Antarmuka Halaman Utama 2

Tabel 4.6 Penjelasan Atribut Perancangan Antarmuka Halaman Utama

No.	Nama Atribut Antarmuka	Jenis Atribut	Kegunaan
1	<i>Entity Combo Box</i>	<i>Form</i>	Menampilkan daftar entitas
2	<i>SubmitButton</i>	<i>Button</i>	Mengeksekusi <i>request form</i>
3	<i>Granularity Table</i>	<i>Table</i>	Menampilkan granularitas waktu dari entitas yang dipilih
4	<i>Relation Table</i>	<i>Table</i>	Menampilkan relasi dari entitas yang dipilih

[Halaman ini sengaja dikosongkan]

BAB V IMPLEMENTASI

Bab ini membahas tentang implementasi dari perancangan sistem yang telah dibuat. Bab ini berisi proses implementasi dari setiap fungsi pada perangkat lunak iStory. Bahasa yang digunakan untuk membangun sistem ini adalah PHP dengan library EasyRDF.

5.1 Implementasi Fungsi

Pada bagian ini dijelaskan secara terperinci mengenai implementasi fungsi-fungsi yang digunakan dalam membangun sistem.

5.1.1 Fungsi *Combo Box*

Fungsi *Combo Box* adalah fungsi yang digunakan untuk menampilkan daftar entitas dalam bentuk *form combo box*. *Form* ini memiliki *method get*. Daftar entitas yang ditampilkan di-*query* dari *file* RDF ontologi yang memiliki tipe *ProperInterval*. Hasil *query* kemudian disimpan sebagai *value* untuk masing-masing entitas. *Form* akan mengirim *value* entitas yang dipilih ke fungsi *get description* ketika tombol *submit* menerima aksi. Implementasi fungsi dropdown select dapat dilihat pada Kode Sumber 5.1

	<code>foreach(\$graph-</code>
1	<code>>allOfType('time:ProperInterval') as \$name) {</code>
	<code>echo "<option</code>
2	<code>value='". \$name. "'>". \$name-</code>
	<code>>get('rdfs:label')."</option>";</code>

Kode Sumber 5.1 Fungsi *Combo Box*

5.1.2 Fungsi *Get Description*

Fungsi *Get Description* adalah fungsi yang digunakan untuk menangkap masukan dari *combo box*. *Value* yang ditangkap digunakan untuk mendapatkan informasi yang

termuat dalam ontologi. Fungsi ini terdiri dari beberapa sub fungsi menurut kegunaannya dalam mengambil informasi dari ontologi.

5.1.2.1 Fungsi *Get Related Entity*

Fungsi ini digunakan untuk mengambil informasi mengenai objek warisan budaya yang terkait dengan entitas terkait yang ditampilkan oleh sistem. Fungsi *get related entity* ditunjukkan oleh Kode Sumber 5.2.

1	<code>\$related = \$ins->get('cidoc:P4i is time-span of');</code>
2	<code>if(\$related){</code>
3	<code>echo '<div class="callout info">';</code>
4	<code>echo 'Entitas Terkait :'.str_replace('http://www.istory.id#', '', \$related->get('rdfs:label')).'
';</code>
5	<code>\$relatedtypes = \$related->types();</code>
6	<code>\$lasttype = end(\$relatedtypes);</code>
7	<code>echo '
Tipe : ';</code>
8	<code>foreach (\$relatedtypes as \$relatedtype)</code>
9	<code>{</code>
9	<code>if(\$relatedtype != \$lasttype)</code>
10	<code>echo \$relatedtype.', ';</code>
11	<code>else echo \$relatedtype;</code>
12	<code>}</code>
13	<code>if(\$related->get('rdfs:comment'))</code>
14	<code>echo '

Deskripsi : '.\$related->get('rdfs:comment').'

';</code>
15	<code>echo '</div>';</code>
16	<code>}</code>

Kode Sumber 5.2 *Get Related Entity*

5.1.2.2 Fungsi *Get Granularity*

Fungsi ini digunakan untuk mengambil informasi waktu berdasarkan granularitas dari suatu entitas. Informasi tanggal, bulan dan tahun dalam ontologi disimpan dalam format *dateTime*. Sedangkan informasi abad disimpan dalam URI. Fungsi ini memecah data yang terkandung dalam format *dateTime* ke dalam format baru yaitu tanggal, bulan dan tahun.

5.1.2.2.1 Granularitas Waktu Mulai

Granularitas dari waktu dimulainya suatu *ProperInterval* diambil dari *object property hasBeginning*, kemudian diambil lagi nilai dari property *inXSDDateTime*. Informasi abad diambil dari property *intervalStarts* dan *intervalDuring*. Implementasi dari fungsi ini ditunjukkan pada Kode Sumber 5.3.

1	<code>\$beginning = \$graph->resource(\$ins->get('time:hasBeginning'));</code>
2	<code>\$begindate = \$beginning->get('time:inXSDDateTime');</code>
3	<code>echo '<p class="label">Awal</p>';</code>
4	<code>if(\$beginning)</code>
5	<code>echo '<div class="large-12 columns">'.str_replace('http://www.istory.id#', "", \$beginning->get('rdfs:label')).'</div>';</code>
6	<code>if(\$begindate){</code>
7	<code>\$date = date_create_from_format('Y-m-d\TH:i:s', \$begindate);</code>
8	
9	<code>if (\$beginning->get('time:unitType')->toString() == "http://www.w3.org/2006/time#unitDay") {</code>

10	<code>echo '<div class="large-4 columns date"><h6 class="subheader">Tanggal</h6><h4>'.date format(\$date,'d').<h4></div>';</code>
11	<code>echo '<div class="large-4 columns date"><h6 class="subheader">Bulan</h6><h4>'.date format(\$date,'M').<h4></div>';</code>
12	<code>echo '<div class="large-4 columns date"><h6 class="subheader">Tahun</h6><h4>'.date format(\$date,'Y').<h4></div>';</code>
13	<code>}</code>
14	
15	<code>elseif (\$beginning- >get('time:unitType')->__toString() == "http://www.w3.org/2006/time#unitMonth<br "){<="" code=""/></code>
16	<code>echo '<div class="large-4 columns date"><h6 class="subheader">Bulan</h6><h4>'.date format(\$date,'M').<h4></div>';</code>
17	<code>echo '<div class="large-4 columns date"><h6 class="subheader">Tahun</h6><h4>'.date format(\$date,'Y').<h4></div>';</code>
18	<code>}</code>
19	
20	<code>elseif(\$beginning- >get('time:unitType')->__toString() == "http://www.w3.org/2006/time#unitYear") {</code>
21	<code>echo '<div class="large-4 columns date"><h6 class="subheader">Tahun</h6><h4>'.date format(\$date,'Y').<h4></div>';</code>
22	<code>}</code>

23	}
24	if (\$centurybegin)
25	echo '<div class="large-12 columns date"><h6 class="subheader">Abad</h6><h4>'. \$centurybegin. '<h4></div> ';

Kode Sumber 5.3 Get Granularitas Waktu Mulai

5.1.2.2.2 Granularitas Waktu Akhir

Granularitas dari waktu berakhirnya suatu *ProperInterval* diambil dari *object property hasEnd*, kemudian diambil lagi nilai dari properti *inXSDDateTime*. Informasi abad diambil dari properti *intervalFinishes* dan *intervalDuring*. Implementasi dari fungsi ini ditunjukkan pada Kode Sumber 5.4.

1	\$send = \$graph->resource(\$ins->get('time:hasEnd'));
2	\$enddate = \$send->get('time:inXSDDateTime');
3	echo '<p class="label">Akhir</p>';
4	if(\$send)
5	echo '<div class="large-12 columns">'.str_replace('http://www.istory.id#', "", \$send->get('rdfs:label')).'</div>';
6	if(\$enddate){
7	\$date = date_create_from_format('Y-m-d\TH:i:s', \$enddate);
8	//unitDay
9	if (\$send->get('time:unitType')->toString() == "http://www.w3.org/2006/time#unitDay") {
10	echo '<div class="large-4 columns date"><h6

	<code>class="subheader">Tanggal</h6><h4>'.date format(\$date,'d').<h4></div>';</code>
11	<code>echo '<div class="large-4 columns date"><h6 class="subheader">Bulan</h6><h4>'.date format(\$date,'M').<h4></div>';</code>
12	<code>echo '<div class="large-4 columns date"><h6 class="subheader">Tahun</h6><h4>'.date format(\$date,'Y').<h4></div>';</code>
13	<code>}</code>
14	<code>//unitMonth</code>
15	<code>elseif (\$end->get('time:unitType')- >__toString() == "http://www.w3.org/2006/time#unitMonth<br "){<="" code=""/></code>
16	<code>echo '<div class="large-4 columns date"><h6 class="subheader">Bulan</h6><h4>'.date format(\$date,'M').<h4></div>';</code>
17	<code>echo '<div class="large-4 columns date"><h6 class="subheader">Tahun</h6><h4>'.date format(\$date,'Y').<h4></div>';</code>
18	<code>}</code>
19	<code>//unitYear</code>
20	<code>elseif (\$end->get('time:unitType')- >__toString() == "http://www.w3.org/2006/time#unitYear") {</code>
21	<code>echo '<div class="large-4 columns date"><h6 class="subheader">Tahun</h6><h4>'.date format(\$date,'Y').<h4></div>';</code>
22	<code>}</code>
23	<code>}</code>
24	<code>if (\$centuryend)</code>

25	<pre>echo '<div class="large-12 columns date"><h6 class="subheader">Abad</h6><h4>'. \$cent uryend.'<h4></div> ';</pre>
----	--

Kode Sumber 5.4 Get Granularitas Waktu Selesai

5.1.3 Fungsi *Get Relations*

Fungsi ini digunakan untuk mendapatkan informasi hubungan antar *TemporalEntity*. Hubungan yang dimaksud yaitu *before*, *after*, *equal*, *intervalBefore*, *intervalAfter*, *intervalMeets*, *intervalMetBy*, *intervalOverlaps*, *intervalOverlappedBy*, *intervalStarts*, *intervalStartedBy*, *intervalDuring*, *intervalContains*, *intervalFinishes*, *intervalFinishedBy*, *intervalEquals*.

5.1.3.1 *Get intervalEquals*

Fungsi ini digunakan untuk mendapatkan entitas yang terjadi bersamaan dengan entitas yang dipilih. Informasi ini diambil dari properti *intervalEquals* yang melekat pada entitas. Implementasi dari fungsi ini ditunjukkan pada Kode Sumber 5.5.

1	<pre>if (\$ins->get('time:intervalEquals')) {</pre>
2	<pre>echo '<tr><td>Interval Equals</td><td>';</pre>
3	<pre>foreach (\$ins->all('time:intervalEquals') as \$equal) {</pre>
4	<pre>echo ''.s tr_replace('http://www.istory.id#', "", \$equal).'
';</pre>
5	<pre>}</pre>
6	<pre>echo '</td></tr>';}</pre>

Kode Sumber 5.5 *Get intervalEquals*

5.1.3.2 *Get intervalBefore*

Fungsi ini digunakan untuk mendapatkan entitas yang terjadi sebelum entitas yang dipilih. Informasi ini diambil dari properti *intervalBefore* yang melekat pada entitas. Implementasi dari fungsi ini ditunjukkan pada Kode Sumber 5.6.

1	<code>if(\$ins->get('time:intervalBefore')){</code>
2	<code>echo '<tr><td>Interval Before</td><td>';</code>
3	<code>foreach (\$ins-></code> <code>>all('time:intervalBefore') as \$before)</code> <code>{</code>
4	<code>echo '<a</code> <code>href="?entity='.urlencode(\$before).'">'.str_replace('http://www.istory.id#', "", \$before).'</code> <code>
';</code>
5	<code>}</code>
6	<code>echo '</td></tr>';</code>
7	<code>}</code>

Kode Sumber 5.6 *Get intervalBefore*

5.1.3.3 *Get intervalAfter*

Fungsi ini digunakan untuk mendapatkan entitas yang terjadi setelah entitas yang dipilih. Informasi ini diambil dari properti *intervalAfter* yang melekat pada entitas. Implementasi dari fungsi ini ditunjukkan pada Kode Sumber 5.7.

Kode Sumber 5.7 *Get intervalAfter*

5.1.3.4 *Get intervalOverlaps*

Fungsi ini digunakan untuk mendapatkan entitas yang di-overlap oleh entitas yang dipilih. Informasi ini diambil dari properti *intervalOverlaps* yang melekat pada entitas. Implementasi dari fungsi ini ditunjukkan pada Kode Sumber 5.8.

1	if(\$ins->get('time:intervalOverlaps')){
2	echo '<tr><td>Interval Overlaps</td><td>';
3	foreach (\$ins->all('time:intervalOverlaps') as \$overlap) {
4	echo ' .str_replace('http://www.istory.id#', "", \$overlap). ';
5	}
6	echo '</td></tr>'; }

Kode Sumber 5.8 Get intervalOverlaps

5.1.3.5 Get intervalOverlappedBy

Fungsi ini digunakan untuk mendapatkan entitas yang meng-*overlap* entitas yang dipilih. Informasi ini diambil dari properti *intervalOverlappedBy* yang melekat pada entitas. Implementasi dari fungsi ini ditunjukkan pada Kode Sumber 5.9.

1	if(\$ins->get('time:intervalOverlappedBy')){
2	echo '<tr><td>Interval OverlappedBy</td><td>';
3	foreach (\$ins->all('time:intervalOverlappedBy') as \$overlapped) {
4	echo ''.str_replace('http://www.istory.id#', "", \$overlapped). ';
5	}
6	echo '</td></tr>'; }

Kode Sumber 5.9 Get intervalOverlappedBy

5.1.3.6 *Get intervalContains*

Fungsi ini digunakan untuk mendapatkan entitas yang terjadi dalam interval waktu entitas yang dipilih. Informasi ini diambil dari properti *intervalContains* yang melekat pada entitas. Implementasi dari fungsi ini ditunjukkan pada Kode Sumber 5.10.

1	<code>if(\$ins->get('time:intervalContains')){</code>
2	<code>echo '<tr><td>Interval Contains</td><td>';</code>
3	<code>foreach (\$ins-> >all('time:intervalContains') as \$contain) {</code>
4	<code>echo ' .str_replace('http://www.istory.id#', "", \$contain).'/
';</code>
5	<code>}</code>
6	<code>echo '</td></tr>';</code>
7	<code>}</code>

Kode Sumber 5.10 *Get intervalContains*

5.1.3.7 *Get intervalDuring*

Fungsi ini digunakan untuk mendapatkan entitas yang didalamnya terjadi interval entitas yang dipilih. Informasi ini diambil dari properti *intervalDuring* yang melekat pada entitas. Implementasi dari fungsi ini ditunjukkan pada Kode Sumber 5.11.

1	<code>if(\$ins->get('time:intervalDuring')){</code>
2	<code>echo '<tr><td>Interval During</td><td>';</code>
3	<code>foreach (\$ins-> >all('time:intervalDuring') as \$during) {</code>
4	<code>echo ''. </code>

	<code>str_replace('http://www.istory.id#', '', \$during).'</code> ;
5	<code>}</code>
6	<code>echo '</td></tr>';</code>
7	<code>}</code>

Kode Sumber 5.11 *Get intervalDuring*

5.1.3.8 *Get intervalMeets*

Fungsi ini digunakan untuk mendapatkan entitas yang bertemu dengan entitas yang dipilih. Informasi ini diambil dari properti *intervalMeets* yang melekat pada entitas. Implementasi dari fungsi ini ditunjukkan pada Kode Sumber 5.12.

1	<code>if(\$ins->get('time:intervalMeets')){</code>
2	<code>echo '<tr><td>Interval Meets</td><td>';</code>
3	<code>foreach (\$ins->all('time:intervalMeets') as \$meet) {</code>
4	<code>echo ''.str_replace('http://www.istory.id#', '', \$meet).'</code> ;
5	<code>}</code>
6	<code>echo '</td></tr>';</code>
7	<code>}</code>

Kode Sumber 5.12 *Get intervalMeets*

5.1.3.9 *Get intervalMetBy*

Fungsi ini digunakan untuk mendapatkan entitas yang ditemui oleh entitas yang dipilih. Informasi ini diambil dari properti *intervalMetby* yang melekat pada entitas. Implementasi dari fungsi ini ditunjukkan pada Kode Sumber 5.13.

1	<code>if(\$ins->get('time:intervalMetBy')){</code>
2	<code>echo '<tr><td>Interval MetBy</td><td>';</code>
3	<code>foreach (\$ins->all('time:intervalMetBy') as \$metby) {</code>

4	<code>echo ''.s tr_replace('http://www.istory.id#', "", \$metby).'
';</code>
5	<code>}</code>
6	<code>echo '</td></tr>';</code>
7	<code>}</code>

Kode Sumber 5.13 *Get intervalMetBy***5.1.3.10 *Get intervalStarts***

Fungsi ini digunakan untuk mendapatkan entitas yang dimulai oleh entitas yang dipilih. Informasi ini diambil dari properti *intervalStarts* yang melekat pada entitas. Implementasi dari fungsi ini ditunjukkan pada Kode Sumber 5.14.

1	<code>if(\$ins->get('time:intervalStarts')){</code>
2	<code>echo '<tr><td>Interval Starts</td><td>';</code>
3	<code>foreach (\$ins-> >all('time:intervalStarts') as \$start) {</code>
4	<code>echo ''.s tr_replace('http://www.istory.id#', "", \$starts).'
';</code>
5	<code>}</code>
6	<code>echo '</td></tr>';</code>
7	<code>}</code>

Kode Sumber 5.14 *Get intervalStarts***5.1.3.11 *Get intervalStartedBy***

Fungsi ini digunakan untuk mendapatkan entitas yang memulai interval entitas yang dipilih. Informasi ini diambil dari properti *intervalStartedBy* yang melekat pada entitas. Implementasi dari fungsi ini ditunjukkan pada Kode Sumber 5.15.

1	if(\$ins->get('time:intervalStartedBy')){
2	echo '<tr><td>Interval StartedBy</td><td>';
3	foreach (\$ins->all('time:intervalStartedBy') as \$startedby) {
4	echo ''.str_replace('http://www.istory.id#', "", \$startedby).' ';
5	}
6	echo '</td></tr>';
7	}

Kode Sumber 5.15 *Get intervalStartedBy*

5.1.3.12 *Get intervalFinishes*

Fungsi ini digunakan untuk mendapatkan entitas yang diakhiri oleh entitas yang dipilih. Informasi ini diambil dari properti *intervalFinishes* yang melekat pada entitas. Implementasi dari fungsi ini ditunjukkan pada Kode Sumber 5.16.

1	if(\$ins->get('time:intervalFinishes')){
2	echo '<tr><td>Interval Finishes</td><td>';
3	foreach (\$ins->all('time:intervalFinishes') as \$finishes) {
4	echo ' ' .str_replace('http://www.istory.id#', "", \$finishes).' ';
5	}
6	echo '</td></tr>'; }

Kode Sumber 5.16 *Get intervalFinishes*

5.1.3.13 *Get intervalFinishedBy*

Fungsi ini digunakan untuk mendapatkan entitas yang mengakhiri interval entitas yang dipilih. Informasi ini diambil dari properti *intervalFinishedBy* yang melekat pada entitas. Implementasi dari fungsi ini ditunjukkan pada Kode Sumber 5.17.

1	<code>if(\$ins-</code> <code>>get('time:intervalFinishedBy')){</code>
2	<code>echo '<tr><td>Interval</code> <code>FinishedBy</td><td>';</code>
3	<code>foreach (\$ins-</code> <code>>all('time:intervalFinishedBy') as</code> <code>\$finishedby) {</code>
4	<code>echo '<a</code> <code>href="?entity='.urlencode(\$finishedby).'</code> <code>">'.str_replace('http://www.istory.id#',</code> <code>"" , \$finishedby).'
';</code>
5	<code>}</code>
6	<code>echo '</td></tr>';</code>
7	<code>}</code>

Kode Sumber 5.17 *Get intervalFinishedBy*

5.2 Implementasi Antarmuka

Implementasi tampilan antarmuka pengguna pada iStory dilakukan dengan menggunakan bahasa HTML dan CSS. Tampilan antarmuka terdiri dari satu halaman utama. Tampilan antarmuka halaman utama dapat dilihat pada Gambar 5.1 dan Gambar 5.2. Gambar 5.2 merupakan tampilan antarmuka awal yaitu ketika belum ada deskripsi entitas temporal yang ditampilkan. Sistem menampilkan *dropdown select* yang berisi daftar entitas temporal yang bisa dipilih. Gambar 5.1 merupakan tampilan antarmuka ketika sistem menampilkan deskripsi entitas yang telah dipilih. Sistem menampilkan deskripsi objek bersejarah yang terkait, deskripsi tanggal dan hubungan antar waktu dengan entitas temporal lain.

PILIH EVENT

Era Kerajaan Singhasari
Submit

DESKRIPSI

0056688 Pecahan Tembikar/ Gerabah

Type : owl:NamedIndividual, cidoc:E1_CRM_Entity, cidoc:E2_Temporal_Entity, cidoc:E4_Period, cidoc:E52_Time-Span, time:Interval, time:ProperInterval, time:TemporalEntity

Entitas Terkait : 0056688 Pecahan Tembikar/ Gerabah

Type : owl:NamedIndividual, cidoc:E1_CRM_Entity, cidoc:E2_Temporal_Entity, cidoc:E70_Thing, cidoc:E77_Persistent_Item

Awal

Awal 0056688 Pecahan Tembikar/ Gerabah

Akhir

Akhir 0056688 Pecahan Tembikar/ Gerabah

Relasi

Property	Value
	Era Kerajaan Singhasari
	Era Ken Arok
	Era Kertanegara
	Era Kerajaan Kadiri
	0056736 Pecahan Keramik/ Piring
	0056739 Pecahan Keramik/ Mangkuk
	0056740 Pecahan Keramik/ Mangkuk

Gambar 5.1 Implementasi Antarmuka Halaman Utama 1

SOERABAJA

PILIH EVENT

Era Kerajaan Singhasari	▼	Submit
Era Kerajaan Singhasari	▲	
Era Ken Arok		
Era Kertanegara		
Era Kerajaan Kadiri		
0056688 Pecahan Tembikar/ Gerabah		
0056733 Pecahan Keramik/ Mangkuk		
0056736 Pecahan Keramik/ Piring		
0056737 Pecahan Gelang		
0056739 Pecahan Keramik/ Mangkuk		
0056740 Pecahan Keramik/ Mangkuk		
0056741 Pecahan Keramik/ Piring		
0056751 Pecahan Keramik/ Piring		
0056788 Pecahan Tembikar/ Gerabah		
Era Kesultanan Samudera Pasai		
0068098 Pecahan Tembikar / Gerabah		
Era Kesultanan Aceh		
Era Ali Mughayat Syah		
Era Alaidin Muhammad Daud Syah		
Era Kesultanan Demak		
Era Raden Patah	▼	

Gambar 5.2 Implementasi Antarmuka Halaman Utama 2

BAB VI

PENGUJIAN DAN EVALUASI

Bab ini membahas pengujian dan evaluasi pada perangkat lunak yang dikembangkan. Pengujian yang dilakukan adalah pengujian model ontologi, kebutuhan fungsionalitas sistem dan pengujian pada studi kasus.

6.1 Lingkungan Pengujian

Lingkungan pengujian sistem pada pengerjaan Tugas Akhir ini dilakukan pada lingkungan dan alat kaku sebagai berikut:

Prosesor	: AMD E-1-1200 CPU @ 1.4GHz
Memori	: 4.00 GB
Jenis <i>Device</i>	: Laptop
Sistem Operasi	: Microsoft Windows 10 Education 32-bit

6.2 Skenario Pengujian

Pada bab ini akan dijelaskan mengenai skenario pengujian yang dilakukan. Pengujian yang dilakukan adalah pengujian terhadap model ontologi dan pengujian terhadap fungsionalitas sistem. Pengujian model ontologi dilakukan Pellet *reasoner* terhadap SWRL rule yang telah dibuat. Pengujian kebutuhan fungsionalitas perangkat lunak menggunakan metode kotak hitam (*black box*). Metode ini menekankan pada hasil uji fungsionalitas sistem yaitu apakah sistem mampu mengeluarkan output sesuai dengan yang diharapkan dari suatu input tertentu.

6.2.1 Pengujian Ontologi

Pengujian ontologi merupakan tahap uji kevalidan ontologi yang telah dibangun sebagai dasar proses pencarian relasi antar tokoh bersejarah di Indonesia. Pengujian dilakukan

secara manual dengan mengecek kebenaran fakta baru yang muncul sebagai tolok ukur keberhasilan pengujian.

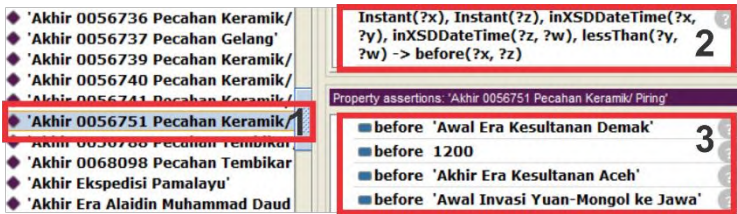
6.2.1.1 Pengujian Kevalidan Relasi *before*

Pada tahap pengujian relasi *before* disediakan masukan berupa *TemporalEntity* yang seharusnya memiliki hubungan *before* dengan *TemporalEntity* lainnya. Tujuannya adalah untuk mengetahui apakah *rule* yang diterapkan benar-benar mampu mendeteksi relasi *before*. Rincian pengujian fitur ini dapat dilihat pada Tabel 6.1.

Tabel 6.1 Pengujian Kevalidan Relasi *before*

ID	TA-UJ.UC0101
Nama	Pengujian kevalidan relasi <i>before</i>
Tujuan Pengujian	Menguji kevalidan rule dan skema ontologi terhadap studi kasus relasi <i>before</i> .
Skenario 1	Melihat <i>inference</i> data individu yang seharusnya memiliki relasi <i>before</i> dan tidak.
Kondisi Awal	Pellet <i>reasoner</i> dalam keadaan mati dan ontologi sudah di impor.
Data Uji	Data uji merupakan individual kelas <i>TemporalEntity</i> yang telah diketahui memiliki relasi <i>before</i> dengan individual lain.
Langkah Pengujian	<ol style="list-style-type: none"> 1. Pengguna memilih individu yang akan diujikan. 2. Pengguna menjalankan <i>reasoner</i> Pellet di Protégé dengan menekan tombol CTRL+R.
Hasil Yang Diharapkan	Pada individu yang diujikan muncul hasil <i>inference</i> berupa <i>object property before</i> .
Hasil Yang Didapat	Pada individu yang seharusnya memiliki relasi <i>before</i> memunculkan hasil <i>inference</i> berupa <i>object property before</i> .
Hasil Pengujian	Berhasil
Kondisi Akhir	Protégé menampilkan individu yang diujikan mengandung <i>inference</i> berupa <i>object property before</i> .

Hasil pengujian kevalidan relasi *before* dapat dilihat pada Gambar 6.1. Dapat dilihat bahwa ketika uji coba dilakukan individu yang seharusnya memiliki relasi *before* dengan individu lain memunculkan data *inference before* sebagai fakta baru. *Pointer 1* menunjukkan individu yang dipakai sebagai uji coba. *Pointer 2* menunjukkan SWRL rule yang dipakai sebagai uji coba. SWRL rule yang digunakan sebagai uji coba merupakan rule yang telah dijelaskan pada Sub bab Pencarian Relasi *before*. Sedangkan *pointer 3* menunjukkan fakta baru yang berupa data *inference* setelah proses *reasoning*. Hasil tersebut membuktikan bahwa uji coba kevalidan relasi *before* telah berhasil.



Gambar 6.1 Pengujian Kevalidan Relasi *before*

6.2.1.2 Pengujian Kevalidan Relasi *after*

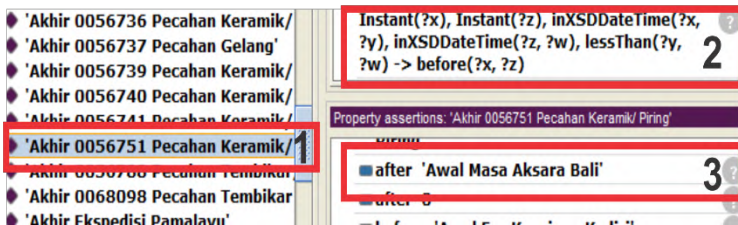
Pada tahap pengujian relasi *after* disediakan masukan berupa *TemporalEntity* yang seharusnya memiliki hubungan *after* dengan *TemporalEntity* lainnya. Tujuannya adalah untuk mengetahui apakah rule yang diterapkan benar-benar mampu mendeteksi relasi *after*. Rincian pengujian fitur ini dapat dilihat pada Tabel 6.2.

Tabel 6.2 Pengujian Kevalidan Relasi *after*

ID	TA-UJ.UC0102
Nama	Pengujian kevalidan relasi <i>after</i>

Tujuan Pengujian	Menguji kevalidan rule dan skema ontologi terhadap studi kasus relasi <i>after</i> .
Skenario 1	Melihat <i>inference</i> data individu yang seharusnya memiliki relasi <i>after</i> dan tidak.
Kondisi Awal	Pellet <i>reasoner</i> dalam keadaan mati dan ontologi sudah di impor.
Data Uji	Data uji merupakan individual kelas <i>TemporalEntity</i> yang telah diketahui memiliki relasi <i>after</i> dengan individual lain.
Langkah Pengujian	<ol style="list-style-type: none"> 1. Pengguna memilih individu yang akan diujikan. 2. Pengguna menjalankan <i>reasoner</i> Pellet di Protégé dengan menekan tombol CTRL+R.
Hasil Yang Diharapkan	Pada individu yang diujikan muncul hasil <i>inference</i> berupa <i>object property after</i> .
Hasil Yang Didapat	Pada individu yang seharusnya memiliki relasi <i>after</i> memunculkan hasil <i>inference</i> berupa <i>object property after</i> .
Hasil Pengujian	Berhasil
Kondisi Akhir	Protégé menampilkan individu yang diujikan mengandung <i>inference</i> berupa <i>object property after</i> .

Hasil pengujian kevalidan relasi *after* dapat dilihat pada Gambar 6.2. Dapat dilihat bahwa ketika uji coba dilakukan individu yang seharusnya memiliki relasi *after* dengan individu lain memunculkan data *inference after* sebagai fakta baru. *Pointer 1* menunjukkan individu yang dipakai sebagai uji coba. *Pointer 2* menunjukkan SWRL rule yang dipakai sebagai uji coba. SWRL rule yang digunakan sebagai uji coba merupakan rule yang telah dijelaskan pada Sub bab Pencarian Relasi *before*. Sedangkan *pointer 3* menunjukkan fakta baru yang berupa data *inference* setelah proses *reasoning*. Hasil tersebut membuktikan bahwa uji coba kevalidan relasi *after* telah berhasil.



Gambar 6.2 Pengujian Kevalidan Relasi *after*

6.2.1.3 Pengujian Kevalidan Relasi *equals*

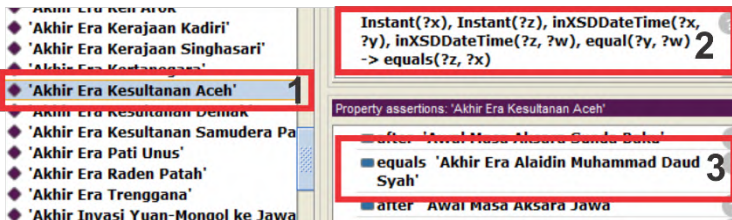
Pada tahap pengujian relasi *equals* disediakan masukan berupa *TemporalEntity* yang seharusnya memiliki hubungan *equals* dengan *TemporalEntity* lainnya. Tujuannya adalah untuk mengetahui apakah *rule* yang diterapkan benar-benar mampu mendeteksi relasi *equals*. Rincian pengujian fitur ini dapat dilihat pada Tabel 6.3.

Tabel 6.3 Pengujian Kevalidan Relasi *equals*

ID	TA-UJ.UC0103
Nama	Pengujian kevalidan relasi <i>equals</i>
Tujuan Pengujian	Menguji kevalidan rule dan skema ontologi terhadap studi kasus relasi <i>equals</i> .
Skenario 1	Melihat <i>inference</i> data individu yang seharusnya memiliki relasi <i>equals</i> dan tidak.
Kondisi Awal	Pellet <i>reasoner</i> dalam keadaan mati dan ontologi sudah di impor.
Data Uji	Data uji merupakan individual kelas <i>TemporalEntity</i> yang telah diketahui memiliki relasi <i>equals</i> dengan individual lain.
Langkah Pengujian	<ol style="list-style-type: none"> 1. Pengguna memilih individu yang akan diujikan. 2. Pengguna menjalankan <i>reasoner</i> Pellet di Protégé dengan menekan tombol CTRL+R.

Hasil Yang Diharapkan	Pada individu yang diujikan muncul hasil <i>inference</i> berupa <i>object property equals</i> .
Hasil Yang Didapat	Pada individu yang seharusnya memiliki relasi <i>equals</i> memunculkan hasil <i>inference</i> berupa <i>object property equals</i> .
Hasil Pengujian	Berhasil
Kondisi Akhir	Protégé menampilkan individu yang diujikan mengandung <i>inference</i> berupa <i>object property equals</i> .

Hasil pengujian kevalidan relasi *equals* dapat dilihat pada Gambar 6.3. Dapat dilihat bahwa ketika uji coba dilakukan individu yang seharusnya memiliki relasi *equals* dengan individu lain memunculkan data *inference equals* sebagai fakta baru. *Pointer 1* menunjukkan individu yang dipakai sebagai uji coba. *Pointer 2* menunjukkan SWRL rule yang dipakai sebagai uji coba. SWRL rule yang digunakan sebagai uji coba merupakan rule yang telah dijelaskan pada Sub bab Pencarian Relasi *equals*. Sedangkan *pointer 3* menunjukkan fakta baru yang berupa data *inference* setelah proses *reasoning*. Hasil tersebut membuktikan bahwa uji coba kevalidan relasi *equals* telah berhasil.



Gambar 6.3 Pengujian Kevalidan Relasi *equals*

6.2.1.4 Pengujian Kevalidan Relasi *intervalMeets*

Pada tahap pengujian relasi *intervalMeets* disediakan masukan berupa *TemporalEntity* yang seharusnya memiliki hubungan *intervalMeets* dengan *TemporalEntity* lainnya.

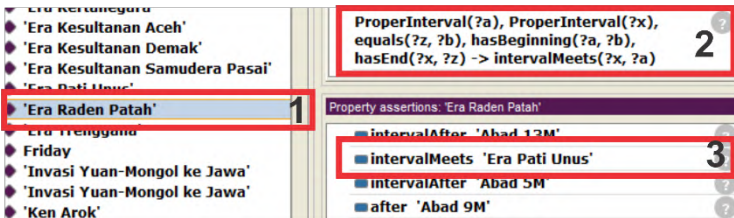
Tujuannya adalah untuk mengetahui apakah *rule* yang diterapkan benar-benar mampu mendeteksi relasi *intervalMeets*. Rincian pengujian fitur ini dapat dilihat pada Tabel 6.4.

Tabel 6.4 Pengujian Kevalidan Relasi *intervalMeets*

ID	TA-UJ.UC0104
Nama	Pengujian kevalidan relasi <i>intervalMeets</i>
Tujuan Pengujian	Menguji kevalidan rule dan skema ontologi terhadap studi kasus relasi <i>intervalMeets</i> .
Skenario 1	Melihat <i>inference</i> data individu yang seharusnya memiliki relasi <i>intervalMeets</i> dan tidak.
Kondisi Awal	Pellet <i>reasoner</i> dalam keadaan mati dan ontologi sudah di impor.
Data Uji	Data uji merupakan individual kelas <i>TemporalEntity</i> yang telah diketahui memiliki relasi <i>intervalMeets</i> dengan individual lain.
Langkah Pengujian	<ol style="list-style-type: none"> 1. Pengguna memilih individu yang akan diujikan. 2. Pengguna menjalankan <i>reasoner</i> Pellet di Protégé dengan menekan tombol CTRL+R.
Hasil Yang Diharapkan	Pada individu yang diujikan muncul hasil <i>inference</i> berupa <i>object property intervalMeets</i> .
Hasil Yang Didapat	Pada individu yang seharusnya memiliki relasi <i>equals</i> memunculkan hasil <i>inference</i> berupa <i>object property intervalMeets</i> .
Hasil Pengujian	Berhasil
Kondisi Akhir	Protégé menampilkan individu yang diujikan mengandung <i>inference</i> berupa <i>object property intervalMeets</i> .

Hasil pengujian kevalidan relasi *intervalMeets* dapat dilihat pada Gambar 6.4. Dapat dilihat bahwa ketika uji coba dilakukan individu yang seharusnya memiliki relasi *intervalMeets* dengan individu lain memunculkan data

inference intervalMeets sebagai fakta baru. *Pointer 1* menunjukkan individu yang dipakai sebagai uji coba. *Pointer 2* menunjukkan SWRL *rule* yang dipakai sebagai uji coba. SWRL *rule* yang digunakan sebagai uji coba merupakan *rule* yang telah dijelaskan pada Sub bab Pencarian Relasi *intervalMeets*. Sedangkan *pointer 3* menunjukkan fakta baru yang berupa data *inference* setelah proses *reasoning*. Hasil tersebut membuktikan bahwa uji coba kevalidan relasi *intervalMeets* telah berhasil.



Gambar 6.4 Pengujian Kevalidan Relasi *intervalMeets*

6.2.1.5 Pengujian Kevalidan Relasi *intervalMetBy*

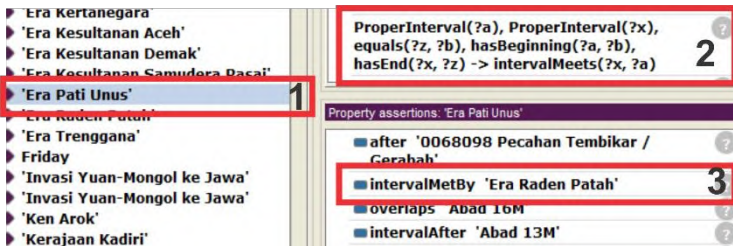
Pada tahap pengujian relasi *intervalMetBy* disediakan masukan berupa *TemporalEntity* yang seharusnya memiliki hubungan *intervalMetBy* dengan *TemporalEntity* lainnya. Tujuannya adalah untuk mengetahui apakah *rule* yang diterapkan benar-benar mampu mendeteksi relasi *intervalMetBy*. Rincian pengujian fitur ini dapat dilihat pada Tabel 6.5.

Tabel 6.5 Pengujian Kevalidan Relasi *intervalMetBy*

ID	TA-UJ.UC0105
Nama	Pengujian kevalidan relasi <i>intervalMetBy</i>
Tujuan Pengujian	Menguji kevalidan rule dan skema ontologi terhadap studi kasus relasi <i>intervalMetBy</i> .

Skenario 1	Melihat <i>inference</i> data individu yang seharusnya memiliki relasi <i>intervalMetBy</i> dan tidak.
Kondisi Awal	Pellet <i>reasoner</i> dalam keadaan mati dan ontologi sudah di impor.
Data Uji	Data uji merupakan individual kelas <i>TemporalEntity</i> yang telah diketahui memiliki relasi <i>intervalMetBy</i> dengan individual lain.
Langkah Pengujian	<ol style="list-style-type: none"> 1. Pengguna memilih individu yang akan diujikan. 2. Pengguna menjalankan <i>reasoner</i> Pellet di Protégé dengan menekan tombol CTRL+R.
Hasil Yang Diharapkan	Pada individu yang diujikan muncul hasil <i>inference</i> berupa <i>object property intervalMetBy</i> .
Hasil Yang Didapat	Pada individu yang seharusnya memiliki relasi <i>equals</i> memunculkan hasil <i>inference</i> berupa <i>object property intervalMetBy</i> .
Hasil Pengujian	Berhasil
Kondisi Akhir	Protégé menampilkan individu yang diujikan mengandung <i>inference</i> berupa <i>object property intervalMetBy</i> .

Hasil pengujian kevalidan relasi *intervalMetBy* dapat dilihat pada Gambar 6.5. Dapat dilihat bahwa ketika uji coba dilakukan individu yang seharusnya memiliki relasi *intervalMetBy* dengan individu lain memunculkan data *inference intervalMetBy* sebagai fakta baru. *Pointer 1* menunjukkan individu yang dipakai sebagai uji coba. *Pointer 2* menunjukkan SWRL rule yang dipakai sebagai uji coba. SWRL rule yang digunakan sebagai uji coba merupakan rule yang telah dijelaskan pada Sub bab Pencarian Relasi *intervalMetBy*. Sedangkan *pointer 3* menunjukkan fakta baru yang berupa data *inference* setelah proses *reasoning*. Hasil tersebut membuktikan bahwa uji coba kevalidan relasi *intervalMetBy* telah berhasil.



Gambar 6.5 Pengujian Kevalidan Relasi *intervalMetBy*

6.2.1.6 Pengujian Kevalidan Relasi *intervalBefore*

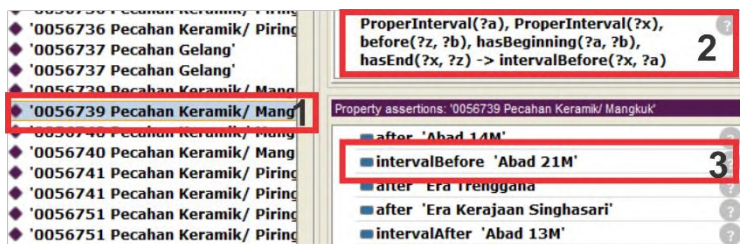
Pada tahap pengujian relasi *intervalBefore* disediakan masukan berupa *TemporalEntity* yang seharusnya memiliki hubungan *intervalBefore* dengan *TemporalEntity* lainnya. Tujuannya adalah untuk mengetahui apakah *rule* yang diterapkan benar-benar mampu mendeteksi relasi *intervalBefore*. Rincian pengujian fitur ini dapat dilihat pada Tabel 6.6.

Tabel 6.6 Pengujian Kevalidan Relasi *intervalBefore*

ID	TA-UJ.UC0106
Nama	Pengujian kevalidan relasi <i>intervalBefore</i>
Tujuan Pengujian	Menguji kevalidan rule dan skema ontologi terhadap studi kasus relasi <i>intervalBefore</i> .
Skenario 1	Melihat <i>inference</i> data individu yang seharusnya memiliki relasi <i>intervalBefore</i> dan tidak.
Kondisi Awal	Pellet <i>reasoner</i> dalam keadaan mati dan ontologi sudah di impor.
Data Uji	Data uji merupakan individual kelas <i>TemporalEntity</i> yang telah diketahui memiliki relasi <i>intervalBefore</i> dengan individual lain.
Langkah Pengujian	<ol style="list-style-type: none"> 1. Pengguna memilih individu yang akan diujikan. 2. Pengguna menjalankan <i>reasoner</i> Pellet di Protégé dengan menekan tombol CTRL+R.

Hasil Yang Diharapkan	Pada individu yang diujikan muncul hasil <i>inference</i> berupa <i>object property intervalBefore</i> .
Hasil Yang Didapat	Pada individu yang seharusnya memiliki relasi <i>equals</i> memunculkan hasil <i>inference</i> berupa <i>object property intervalBefore</i> .
Hasil Pengujian	Berhasil
Kondisi Akhir	Protégé menampilkan individu yang diujikan mengandung <i>inference</i> berupa <i>object property intervalBefore</i> .

Hasil pengujian kevalidan relasi *intervalBefore* dapat dilihat pada Gambar 6.6. Dapat dilihat bahwa ketika uji coba dilakukan individu yang seharusnya memiliki relasi *intervalBefore* dengan individu lain memunculkan data *inference intervalBefore* sebagai fakta baru. *Pointer 1* menunjukkan individu yang dipakai sebagai uji coba. *Pointer 2* menunjukkan SWRL rule yang dipakai sebagai uji coba. SWRL rule yang digunakan sebagai uji coba merupakan rule yang telah dijelaskan pada Sub bab Pencarian Relasi *intervalBefore*. Sedangkan *pointer 3* menunjukkan fakta baru yang berupa data *inference* setelah proses *reasoning*. Hasil tersebut membuktikan bahwa uji coba kevalidan relasi *intervalBefore* telah berhasil.



Gambar 6.6 Pengujian Kevalidan Relasi *intervalBefore*

6.2.1.7 Pengujian Kevalidan Relasi *intervalAfter*

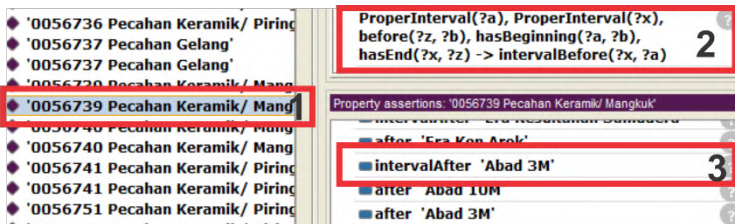
Pada tahap pengujian relasi *intervalAfter* disediakan masukan berupa *TemporalEntity* yang seharusnya memiliki hubungan *intervalAfter* dengan *TemporalEntity* lainnya. Tujuannya adalah untuk mengetahui apakah *rule* yang diterapkan benar-benar mampu mendeteksi relasi *intervalAfter*. Rincian pengujian fitur ini dapat dilihat pada

Tabel 6.7.

Tabel 6.7 Pengujian Kevalidan Relasi *intervalAfter*

ID	TA-UJ.UC0107
Nama	Pengujian kevalidan relasi <i>intervalAfter</i>
Tujuan Pengujian	Menguji kevalidan rule dan skema ontologi terhadap studi kasus relasi <i>intervalAfter</i> .
Skenario 1	Melihat <i>inference</i> data individu yang seharusnya memiliki relasi <i>intervalAfter</i> dan tidak.
Kondisi Awal	Pellet <i>reasoner</i> dalam keadaan mati dan ontologi sudah di impor.
Data Uji	Data uji merupakan individual kelas <i>TemporalEntity</i> yang telah diketahui memiliki relasi <i>intervalAfter</i> dengan individual lain.
Langkah Pengujian	<ol style="list-style-type: none"> 1. Pengguna memilih individu yang akan diujikan. 2. Pengguna menjalankan <i>reasoner</i> Pellet di Protégé dengan menekan tombol CTRL+R.
Hasil Yang Diharapkan	Pada individu yang diujikan muncul hasil <i>inference</i> berupa <i>object property intervalAfter</i> .
Hasil Yang Didapat	Pada individu yang seharusnya memiliki relasi <i>equals</i> memunculkan hasil <i>inference</i> berupa <i>object property intervalAfter</i> .
Hasil Pengujian	Berhasil
Kondisi Akhir	Protégé menampilkan individu yang diujikan mengandung <i>inference</i> berupa <i>object property intervalAfter</i> .

Hasil pengujian kevalidan relasi *intervalAfter* dapat dilihat pada Gambar 6.7. Dapat dilihat bahwa ketika uji coba dilakukan individu yang seharusnya memiliki relasi *intervalAfter* dengan individu lain memunculkan data *inference intervalAfter* sebagai fakta baru. *Pointer 1* menunjukkan individu yang dipakai sebagai uji coba. *Pointer 2* menunjukkan SWRL rule yang dipakai sebagai uji coba. SWRL rule yang digunakan sebagai uji coba merupakan rule yang telah dijelaskan pada Sub bab Pencarian Relasi *intervalAfter*. Sedangkan *pointer 3* menunjukkan fakta baru yang berupa data *inference* setelah proses *reasoning*. Hasil tersebut membuktikan bahwa uji coba kevalidan relasi *intervalAfter* telah berhasil.



Gambar 6.7 Pengujian Kevalidan Relasi *intervalAfter*

6.2.1.8 Pengujian Kevalidan Relasi *intervalDuring*

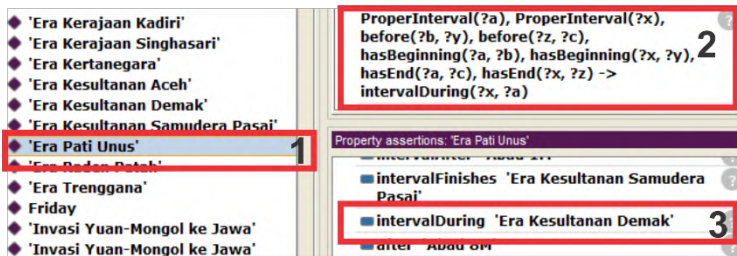
Pada tahap pengujian relasi *intervalDuring* disediakan masukan berupa *TemporalEntity* yang seharusnya memiliki hubungan *intervalDuring* dengan *TemporalEntity* lainnya. Tujuannya adalah untuk mengetahui apakah rule yang diterapkan benar-benar mampu mendeteksi relasi *intervalDuring*. Rincian pengujian fitur ini dapat dilihat pada Tabel 6.8.

Tabel 6.8 Pengujian Kevalidan Relasi *intervalDuring*

ID	TA-UJ.UC0108
Nama	Pengujian kevalidan relasi <i>intervalDuring</i>

Tujuan Pengujian	Menguji kevalidan rule dan skema ontologi terhadap studi kasus relasi <i>intervalDuring</i> .
Skenario 1	Melihat <i>inference</i> data individu yang seharusnya memiliki relasi <i>intervalDuring</i> dan tidak.
Kondisi Awal	Pellet <i>reasoner</i> dalam keadaan mati dan ontologi sudah di impor.
Data Uji	Data uji merupakan individual kelas <i>TemporalEntity</i> yang telah diketahui memiliki relasi <i>intervalDuring</i> dengan individual lain.
Langkah Pengujian	<ol style="list-style-type: none"> 1. Pengguna memilih individu yang akan diujikan. 2. Pengguna menjalankan <i>reasoner</i> Pellet di Protégé dengan menekan tombol CTRL+R.
Hasil Yang Diharapkan	Pada individu yang diujikan muncul hasil <i>inference</i> berupa <i>object property intervalDuring</i> .
Hasil Yang Didapat	Pada individu yang seharusnya memiliki relasi <i>equals</i> memunculkan hasil <i>inference</i> berupa <i>object property intervalDuring</i> .
Hasil Pengujian	Berhasil
Kondisi Akhir	Protégé menampilkan individu yang diujikan mengandung <i>inference</i> berupa <i>object property intervalDuring</i> .

Hasil pengujian kevalidan relasi *intervalDuring* dapat dilihat pada Gambar 6.8. Dapat dilihat bahwa ketika uji coba dilakukan individu yang seharusnya memiliki relasi *intervalDuring* dengan individu lain memunculkan data *inference intervalDuring* sebagai fakta baru. *Pointer* 1 menunjukkan individu yang dipakai sebagai uji coba. *Pointer* 2 menunjukkan SWRL *rule* yang dipakai sebagai uji coba. SWRL *rule* yang digunakan sebagai uji coba merupakan *rule* yang telah dijelaskan pada Sub bab Pencarian Relasi *intervalDuring*. Sedangkan *pointer* 3 menunjukkan fakta baru yang berupa data *inference* setelah proses *reasoning*. Hasil tersebut membuktikan bahwa uji coba kevalidan relasi *intervalDuring* telah berhasil.



Gambar 6.8 Pengujian Kevalidan Relasi *intervalDuring*

6.2.1.9 Pengujian Kevalidan Relasi *intervalContains*

Pada tahap pengujian relasi *intervalContains* disediakan masukan berupa *TemporalEntity* yang seharusnya memiliki hubungan *intervalContains* dengan *TemporalEntity* lainnya. Tujuannya adalah untuk mengetahui apakah *rule* yang diterapkan benar-benar mampu mendeteksi relasi *intervalContains*. Rincian pengujian fitur ini dapat dilihat pada Tabel 6.9.

Tabel 6.9 Pengujian Kevalidan Relasi *intervalContains*

ID	TA-UJ.UC0109
Nama	Pengujian kevalidan relasi <i>intervalContains</i>
Tujuan Pengujian	Menguji kevalidan rule dan skema ontologi terhadap studi kasus relasi <i>intervalContains</i> .
Skenario 1	Melihat <i>inference</i> data individu yang seharusnya memiliki relasi <i>intervalContains</i> dan tidak.
Kondisi Awal	Pellet <i>reasoner</i> dalam keadaan mati dan ontologi sudah di impor.
Data Uji	Data uji merupakan individual kelas <i>TemporalEntity</i> yang telah diketahui memiliki relasi <i>intervalContains</i> dengan individual lain.
Langkah Pengujian	<ol style="list-style-type: none"> 1. Pengguna memilih individu yang akan diujikan. 2. Pengguna menjalankan <i>reasoner</i> Pellet di Protégé dengan menekan tombol CTRL+R.

Hasil Yang Diharapkan	Pada individu yang diujikan muncul hasil <i>inference</i> berupa <i>object property intervalContains</i> .
Hasil Yang Didapat	Pada individu yang seharusnya memiliki relasi <i>equals</i> memunculkan hasil <i>inference</i> berupa <i>object property intervalContains</i> .
Hasil Pengujian	Berhasil
Kondisi Akhir	Protégé menampilkan individu yang diujikan mengandung <i>inference</i> berupa <i>object property intervalContains</i> .

Hasil pengujian kevalidan relasi *intervalContains* dapat dilihat pada Gambar 6.9. Dapat dilihat bahwa ketika uji coba dilakukan individu yang seharusnya memiliki relasi *intervalContains* dengan individu lain memunculkan data *inference intervalContains* sebagai fakta baru. *Pointer 1* menunjukkan individu yang dipakai sebagai uji coba. *Pointer 2* menunjukkan SWRL rule yang dipakai sebagai uji coba. SWRL rule yang digunakan sebagai uji coba merupakan rule yang telah dijelaskan pada Sub bab Pencarian Relasi *intervalContains*. Sedangkan *pointer 3* menunjukkan fakta baru yang berupa data *inference* setelah proses *reasoning*. Hasil tersebut membuktikan bahwa uji coba kevalidan relasi *intervalContains* telah berhasil.

The screenshot displays the Protégé software interface during a reasoning test. On the left, a list of individuals includes 'Era Kesultanan Aceh', which is highlighted with a red box and labeled '1'. On the right, the SWRL rule editor shows a rule for *intervalDuring* based on *ProperInterval* and *before* conditions, highlighted with a red box and labeled '2'. Below the rule, the 'Property assertions' for 'Era Kesultanan Aceh' are shown, including 'intervalAfter Era Ken Arok' and 'intervalContains "0056741 Pecahan Keramik/ Piring"', with the latter highlighted by a red box and labeled '3'.

Gambar 6.9 Pengujian Kevalidan Relasi *intervalContains*

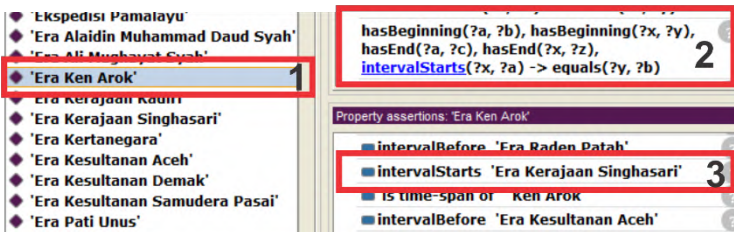
6.2.1.10 Pengujian Kevalidan Relasi *intervalStarts*

Pada tahap pengujian relasi *intervalStarts* disediakan masukan berupa *TemporalEntity* yang seharusnya memiliki hubungan *intervalStarts* dengan *TemporalEntity* lainnya. Tujuannya adalah untuk mengetahui apakah *rule* yang diterapkan benar-benar mampu mendeteksi relasi *intervalStarts*. Rincian pengujian fitur ini dapat dilihat pada Tabel 6.10.

Tabel 6.10 Pengujian Kevalidan Relasi *intervalStarts*

ID	TA-UJ.UC0110
Nama	Pengujian kevalidan relasi <i>intervalStarts</i>
Tujuan Pengujian	Menguji kevalidan rule dan skema ontologi terhadap studi kasus relasi <i>intervalStarts</i> .
Skenario 1	Melihat <i>inference</i> data individu yang seharusnya memiliki relasi <i>intervalStarts</i> dan tidak.
Kondisi Awal	Pellet <i>reasoner</i> dalam keadaan mati dan ontologi sudah di impor.
Data Uji	Data uji merupakan individual kelas <i>TemporalEntity</i> yang telah diketahui memiliki relasi <i>intervalStarts</i> dengan individual lain.
Langkah Pengujian	<ol style="list-style-type: none"> 1. Pengguna memilih individu yang akan diujikan. 2. Pengguna menjalankan <i>reasoner</i> Pellet di Protégé dengan menekan tombol CTRL+R.
Hasil Yang Diharapkan	Pada individu yang diujikan muncul hasil <i>inference</i> berupa <i>object property intervalStarts</i> .
Hasil Yang Didapat	Pada individu yang seharusnya memiliki relasi <i>equals</i> memunculkan hasil <i>inference</i> berupa <i>object property intervalStarts</i> .
Hasil Pengujian	Berhasil
Kondisi Akhir	Protégé menampilkan individu yang diujikan mengandung <i>inference</i> berupa <i>object property intervalStarts</i> .

Hasil pengujian kevalidan relasi *intervalStarts* dapat dilihat pada Gambar 6.10. Dapat dilihat bahwa ketika uji coba dilakukan individu yang seharusnya memiliki relasi *intervalStarts* dengan individu lain memunculkan data *inference intervalStarts* sebagai fakta baru. *Pointer 1* menunjukkan individu yang dipakai sebagai uji coba. *Pointer 2* menunjukkan SWRL rule yang dipakai sebagai uji coba. SWRL rule yang digunakan sebagai uji coba merupakan rule yang telah dijelaskan pada Sub bab Pencarian Relasi *intervalStarts*. Sedangkan *pointer 3* menunjukkan fakta baru yang berupa data *inference* setelah proses *reasoning*. Hasil tersebut membuktikan bahwa uji coba kevalidan relasi *intervalStarts* telah berhasil.



Gambar 6.10 Pengujian Kevalidan Relasi *intervalStarts*

6.2.1.11 Pengujian Kevalidan Relasi *intervalStartedBy*

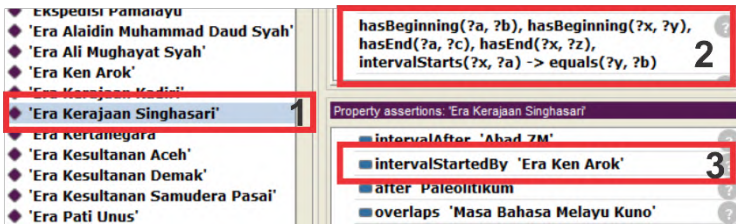
Pada tahap pengujian relasi *intervalStartedBy* disediakan masukan berupa *TemporalEntity* yang seharusnya memiliki hubungan *intervalStartedBy* dengan *TemporalEntity* lainnya. Tujuannya adalah untuk mengetahui apakah rule yang diterapkan benar-benar mampu mendeteksi relasi *intervalStartedBy*. Rincian pengujian fitur ini dapat dilihat pada Tabel 6.11.

Tabel 6.11 Pengujian Kevalidan Relasi *intervalStartedBy*

ID	TA-UJ.UC0111
Nama	Pengujian kevalidan relasi <i>intervalStartedBy</i>

Tujuan Pengujian	Menguji kevalidan rule dan skema ontologi terhadap studi kasus relasi <i>intervalStartedBy</i> .
Skenario 1	Melihat <i>inference</i> data individu yang seharusnya memiliki relasi <i>intervalStartedBy</i> dan tidak.
Kondisi Awal	Pellet <i>reasoner</i> dalam keadaan mati dan ontologi sudah di impor.
Data Uji	Data uji merupakan individual kelas <i>TemporalEntity</i> yang telah diketahui memiliki relasi <i>intervalStartedBy</i> dengan individual lain.
Langkah Pengujian	<ol style="list-style-type: none"> 1. Pengguna memilih individu yang akan diujikan. 2. Pengguna menjalankan <i>reasoner</i> Pellet di Protégé dengan menekan tombol CTRL+R.
Hasil Yang Diharapkan	Pada individu yang diujikan muncul hasil <i>inference</i> berupa <i>object property intervalStartedBy</i> .
Hasil Yang Didapat	Pada individu yang seharusnya memiliki relasi <i>equals</i> memunculkan hasil <i>inference</i> berupa <i>object property intervalStartedBy</i> .
Hasil Pengujian	Berhasil
Kondisi Akhir	Protégé menampilkan individu yang diujikan mengandung <i>inference</i> berupa <i>object property intervalStartedBy</i> .

Hasil pengujian kevalidan relasi *intervalStartedBy* dapat dilihat pada Gambar 6.11. Dapat dilihat bahwa ketika uji coba dilakukan individu yang seharusnya memiliki relasi *intervalStartedBy* dengan individu lain memunculkan data *inference intervalStartedBy* sebagai fakta baru. *Pointer 1* menunjukkan individu yang dipakai sebagai uji coba. *Pointer 2* menunjukkan SWRL rule yang dipakai sebagai uji coba. SWRL rule yang digunakan sebagai uji coba merupakan rule yang telah dijelaskan pada Sub bab Pencarian Relasi *intervalStartedBy*. Sedangkan *pointer 3* menunjukkan fakta baru yang berupa data *inference* setelah proses *reasoning*. Hasil tersebut membuktikan bahwa uji coba kevalidan relasi *intervalStartedBy* telah berhasil.



Gambar 6.11 Pengujian Kevalidan Relasi *intervalStartedBy*

6.2.1.12 Pengujian Kevalidan Relasi *intervalFinishes*

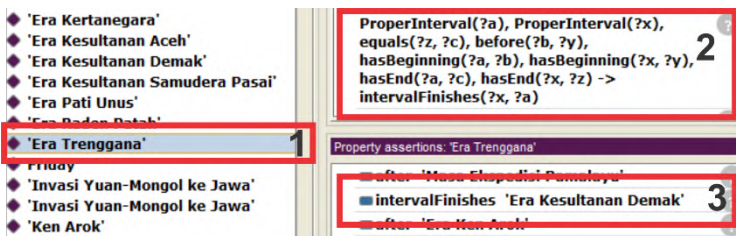
Pada tahap pengujian relasi *intervalFinishes* disediakan masukan berupa *TemporalEntity* yang seharusnya memiliki hubungan *intervalFinishes* dengan *TemporalEntity* lainnya. Tujuannya adalah untuk mengetahui apakah *rule* yang diterapkan benar-benar mampu mendeteksi relasi *intervalFinishes*. Rincian pengujian fitur ini dapat dilihat pada Tabel 6.12.

Tabel 6.12 Pengujian Kevalidan Relasi *intervalFinishes*

ID	TA-UJ.UC0112
Nama	Pengujian kevalidan relasi <i>intervalFinishes</i>
Tujuan Pengujian	Menguji kevalidan rule dan skema ontologi terhadap studi kasus relasi <i>intervalFinishes</i> .
Skenario 1	Melihat <i>inference</i> data individu yang seharusnya memiliki relasi <i>intervalFinishes</i> dan tidak.
Kondisi Awal	Pellet <i>reasoner</i> dalam keadaan mati dan ontologi sudah di impor.
Data Uji	Data uji merupakan individual kelas <i>TemporalEntity</i> yang telah diketahui memiliki relasi <i>intervalFinishes</i> dengan individual lain.
Langkah Pengujian	<ol style="list-style-type: none"> 1. Pengguna memilih individu yang akan diujikan. 2. Pengguna menjalankan <i>reasoner</i> Pellet di Protégé dengan menekan tombol CTRL+R.

Hasil Yang Diharapkan	Pada individu yang diujikan muncul hasil <i>inference</i> berupa <i>object property intervalFinishes</i> .
Hasil Yang Didapat	Pada individu yang seharusnya memiliki relasi <i>equals</i> memunculkan hasil <i>inference</i> berupa <i>object property intervalFinishes</i> .
Hasil Pengujian	Berhasil
Kondisi Akhir	Protégé menampilkan individu yang diujikan mengandung <i>inference</i> berupa <i>object property intervalFinishes</i> .

Hasil pengujian kevalidan relasi *intervalFinishes* dapat dilihat pada Gambar 6.12. Dapat dilihat bahwa ketika uji coba dilakukan individu yang seharusnya memiliki relasi *intervalFinishes* dengan individu lain memunculkan data *inference intervalFinishes* sebagai fakta baru. *Pointer 1* menunjukkan individu yang dipakai sebagai uji coba. *Pointer 2* menunjukkan SWRL rule yang dipakai sebagai uji coba. SWRL rule yang digunakan sebagai uji coba merupakan rule yang telah dijelaskan pada Sub bab Pencarian Relasi *intervalFinishes*. Sedangkan *pointer 3* menunjukkan fakta baru yang berupa data *inference* setelah proses *reasoning*. Hasil tersebut membuktikan bahwa uji coba kevalidan relasi *intervalFinishes* telah berhasil.



Gambar 6.12 Pengujian Kevalidan Relasi *intervalFinishes*

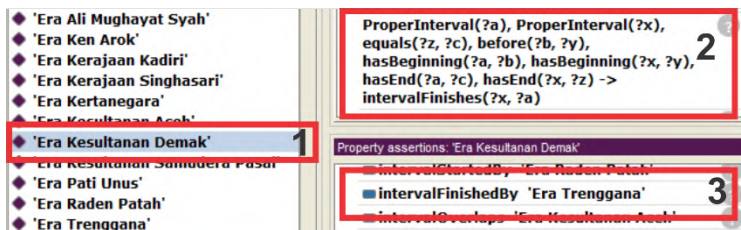
6.2.1.13 Pengujian Kevalidan Relasi *intervalFinishedBy*

Pada tahap pengujian relasi *intervalFinishedBy* disediakan masukan berupa *TemporalEntity* yang seharusnya memiliki hubungan *intervalFinishedBy* dengan *TemporalEntity* lainnya. Tujuannya adalah untuk mengetahui apakah *rule* yang diterapkan benar-benar mampu mendeteksi relasi *intervalFinishedBy*. Rincian pengujian fitur ini dapat dilihat pada Tabel 6.13.

Tabel 6.13 Pengujian Kevalidan Relasi *intervalFinishedBy*

ID	TA-UJ.UC0113
Nama	Pengujian kevalidan relasi <i>intervalFinishedBy</i>
Tujuan Pengujian	Menguji kevalidan rule dan skema ontologi terhadap studi kasus relasi <i>intervalFinishedBy</i> .
Skenario 1	Melihat <i>inference</i> data individu yang seharusnya memiliki relasi <i>intervalFinishedBy</i> dan tidak.
Kondisi Awal	Pellet <i>reasoner</i> dalam keadaan mati dan ontologi sudah di impor.
Data Uji	Data uji merupakan individual kelas <i>TemporalEntity</i> yang telah diketahui memiliki relasi <i>intervalFinishedBy</i> dengan individual lain.
Langkah Pengujian	<ol style="list-style-type: none"> 1. Pengguna memilih individu yang akan diujikan. 2. Pengguna menjalankan <i>reasoner</i> Pellet di Protégé dengan menekan tombol CTRL+R.
Hasil Yang Diharapkan	Pada individu yang diujikan muncul hasil <i>inference</i> berupa <i>object property intervalFinishedBy</i> .
Hasil Yang Didapat	Pada individu yang seharusnya memiliki relasi <i>equals</i> memunculkan hasil <i>inference</i> berupa <i>object property intervalFinishedBy</i> .
Hasil Pengujian	Berhasil
Kondisi Akhir	Protégé menampilkan individu yang diujikan mengandung <i>inference</i> berupa <i>object property intervalFinishedBy</i> .

Hasil pengujian kevalidan relasi *intervalFinishedBy* dapat dilihat pada Gambar 6.13. Dapat dilihat bahwa ketika uji coba dilakukan individu yang seharusnya memiliki relasi *intervalFinishedBy* dengan individu lain memunculkan data *inference intervalFinishedBy* sebagai fakta baru. *Pointer 1* menunjukkan individu yang dipakai sebagai uji coba. *Pointer 2* menunjukkan SWRL rule yang dipakai sebagai uji coba. SWRL rule yang digunakan sebagai uji coba merupakan rule yang telah dijelaskan pada Sub bab Pencarian Relasi *intervalFinishedBy*. Sedangkan *pointer 3* menunjukkan fakta baru yang berupa data *inference* setelah proses *reasoning*. Hasil tersebut membuktikan bahwa uji coba kevalidan relasi *intervalFinishedBy* telah berhasil.



Gambar 6.13 Pengujian Kevalidan Relasi *intervalFinishedBy*

6.2.1.14 Pengujian Kevalidan Relasi *intervalOverlaps*

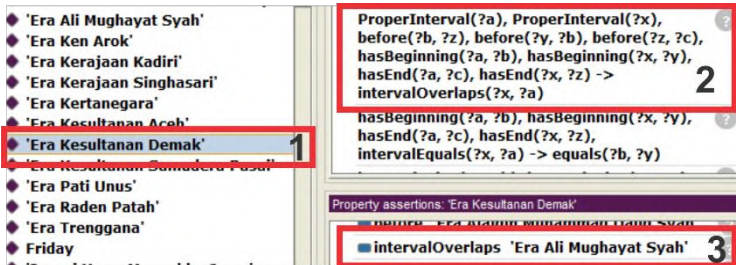
Pada tahap pengujian relasi *intervalOverlaps* disediakan masukan berupa *TemporalEntity* yang seharusnya memiliki hubungan *intervalOverlaps* dengan *TemporalEntity* lainnya. Tujuannya adalah untuk mengetahui apakah rule yang diterapkan benar-benar mampu mendeteksi relasi *intervalOverlaps*. Rincian pengujian fitur ini dapat dilihat pada Tabel 6.14.

Tabel 6.14 Pengujian Kevalidan Relasi *intervalOverlaps*

ID	TA-UJ.UC0114
Nama	Pengujian kevalidan relasi <i>intervalOverlaps</i>
Tujuan Pengujian	Menguji kevalidan rule dan skema ontologi terhadap studi kasus relasi <i>intervalOverlaps</i> .
Skenario 1	Melihat <i>inference</i> data individu yang seharusnya memiliki relasi <i>intervalOverlaps</i> dan tidak.
Kondisi Awal	Pellet <i>reasoner</i> dalam keadaan mati dan ontologi sudah di impor.
Data Uji	Data uji merupakan individual kelas <i>TemporalEntity</i> yang telah diketahui memiliki relasi <i>intervalOverlaps</i> dengan individual lain.
Langkah Pengujian	<ol style="list-style-type: none"> 1. Pengguna memilih individu yang akan diujikan. 2. Pengguna menjalankan <i>reasoner</i> Pellet di Protégé dengan menekan tombol CTRL+R.
Hasil Yang Diharapkan	Pada individu yang diujikan muncul hasil <i>inference</i> berupa <i>object property intervalOverlaps</i> .
Hasil Yang Didapat	Pada individu yang seharusnya memiliki relasi <i>equals</i> memunculkan hasil <i>inference</i> berupa <i>object property intervalOverlaps</i> .
Hasil Pengujian	Berhasil
Kondisi Akhir	Protégé menampilkan individu yang diujikan mengandung <i>inference</i> berupa <i>object property intervalOverlaps</i> .

Hasil pengujian kevalidan relasi *intervalOverlaps* dapat dilihat pada Gambar 6.14. Dapat dilihat bahwa ketika uji coba dilakukan individu yang seharusnya memiliki relasi *intervalOverlaps* dengan individu lain memunculkan data *inference intervalOverlaps* sebagai fakta baru. *Pointer* 1 menunjukkan individu yang dipakai sebagai uji coba. *Pointer* 2 menunjukkan SWRL rule yang dipakai sebagai uji coba. SWRL rule yang digunakan sebagai uji coba merupakan rule yang telah dijelaskan pada Sub bab Pencarian Relasi

intervalOverlaps. Sedangkan *pointer* 3 menunjukkan fakta baru yang berupa data *inference* setelah proses *reasoning*. Hasil tersebut membuktikan bahwa uji coba kevalidan relasi *intervalOverlaps* telah berhasil.



Gambar 6.14 Pengujian Kevalidan Relasi *intervalOverlaps*

6.2.1.15 Pengujian Kevalidan Relasi *intervalOverlappedBy*

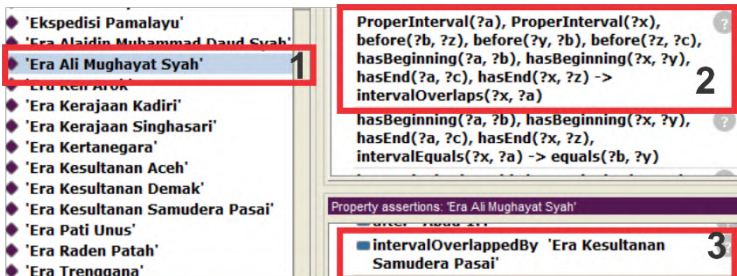
Pada tahap pengujian relasi *intervalOverlappedBy* disediakan masukan berupa *TemporalEntity* yang seharusnya memiliki hubungan *intervalOverlappedBy* dengan *TemporalEntity* lainnya. Tujuannya adalah untuk mengetahui apakah *rule* yang diterapkan benar-benar mampu mendeteksi relasi *intervalOverlappedBy*. Rincian pengujian fitur ini dapat dilihat pada Tabel 6.15.

Tabel 6.15 Pengujian Kevalidan Relasi *intervalOverlappedBy*

ID	TA-UJ.UC0115
Nama	Pengujian kevalidan relasi <i>intervalOverlappedBy</i>
Tujuan Pengujian	Menguji kevalidan rule dan skema ontologi terhadap studi kasus relasi <i>intervalOverlappedBy</i> .
Skenario 1	Melihat <i>inference</i> data individu yang seharusnya memiliki relasi <i>intervalOverlappedBy</i> dan tidak.
Kondisi Awal	Pellet <i>reasoner</i> dalam keadaan mati dan ontologi sudah di impor.

Data Uji	Data uji merupakan individual kelas <i>TemporalEntity</i> yang telah diketahui memiliki relasi <i>intervalOverlappedBy</i> dengan individual lain.
Langkah Pengujian	<ol style="list-style-type: none"> 1. Pengguna memilih individu yang akan diujikan. 2. Pengguna menjalankan <i>reasoner</i> Pellet di Protégé dengan menekan tombol CTRL+R.
Hasil Yang Diharapkan	Pada individu yang diujikan muncul hasil <i>inference</i> berupa <i>object property intervalOverlappedBy</i> .
Hasil Yang Didapat	Pada individu yang seharusnya memiliki relasi <i>equals</i> memunculkan hasil <i>inference</i> berupa <i>object property intervalOverlappedBy</i> .
Hasil Pengujian	Berhasil
Kondisi Akhir	Protégé menampilkan individu yang diujikan mengandung <i>inference</i> berupa <i>object property intervalOverlappedBy</i> .

Hasil pengujian kevalidan relasi *intervalOverlappedBy* dapat dilihat pada Gambar 6.15. Dapat dilihat bahwa ketika uji coba dilakukan individu yang seharusnya memiliki relasi *intervalOverlappedBy* dengan individu lain memunculkan data *inference intervalOverlappedBy* sebagai fakta baru. *Pointer* 1 menunjukkan individu yang dipakai sebagai uji coba. *Pointer* 2 menunjukkan SWRL *rule* yang dipakai sebagai uji coba. SWRL *rule* yang digunakan sebagai uji coba merupakan *rule* yang telah dijelaskan pada Sub bab Pencarian Relasi *intervalOverlappedBy*. Sedangkan *pointer* 3 menunjukkan fakta baru yang berupa data *inference* setelah proses *reasoning*. Hasil tersebut membuktikan bahwa uji coba kevalidan relasi *intervalOverlappedBy* telah berhasil.



Gambar 6.15 Pengujian Kevalidan Relasi *intervalOverlappedBy*

6.2.1.16 Pengujian Kevalidan Relasi *intervalEquals*

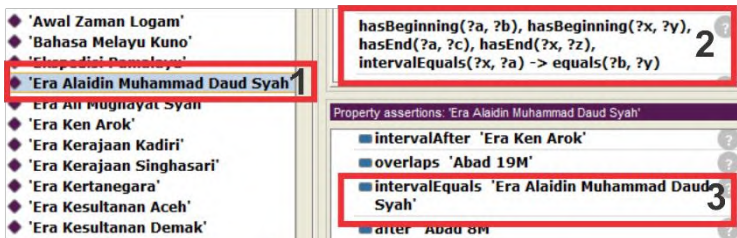
Pada tahap pengujian relasi *intervalEquals* disediakan masukan berupa *TemporalEntity* yang seharusnya memiliki hubungan *intervalEquals* dengan *TemporalEntity* lainnya. Tujuannya adalah untuk mengetahui apakah *rule* yang diterapkan benar-benar mampu mendeteksi relasi *intervalEquals*. Rincian pengujian fitur ini dapat dilihat pada Tabel 6.16.

Tabel 6.16 Pengujian Kevalidan Relasi *intervalEquals*

ID	TA-UJ.UC0116
Nama	Pengujian kevalidan relasi <i>intervalEquals</i>
Tujuan Pengujian	Menguji kevalidan rule dan skema ontologi terhadap studi kasus relasi <i>intervalEquals</i> .
Skenario 1	Melihat <i>inference</i> data individu yang seharusnya memiliki relasi <i>intervalEquals</i> dan tidak.
Kondisi Awal	Pellet <i>reasoner</i> dalam keadaan mati dan ontologi sudah di impor.
Data Uji	Data uji merupakan individual kelas <i>TemporalEntity</i> yang telah diketahui memiliki relasi <i>intervalEquals</i> dengan individual lain.
Langkah Pengujian	1. Pengguna memilih individu yang akan diujikan.

	2. Pengguna menjalankan <i>reasoner</i> Pellet di Protégé dengan menekan tombol CTRL+R.
Hasil Yang Diharapkan	Pada individu yang diujikan muncul hasil <i>inference</i> berupa <i>object property intervalEquals</i> .
Hasil Yang Didapat	Pada individu yang seharusnya memiliki relasi <i>equals</i> memunculkan hasil <i>inference</i> berupa <i>object property intervalEquals</i> .
Hasil Pengujian	Berhasil
Kondisi Akhir	Protégé menampilkan individu yang diujikan mengandung <i>inference</i> berupa <i>object property intervalEquals</i> .

Hasil pengujian kevalidan relasi *intervalEquals* dapat dilihat pada Gambar 6.16. Dapat dilihat bahwa ketika uji coba dilakukan individu yang seharusnya memiliki relasi *intervalEquals* dengan individu lain memunculkan data *inference intervalEquals* sebagai fakta baru. *Pointer* 1 menunjukkan individu yang dipakai sebagai uji coba. *Pointer* 2 menunjukkan SWRL rule yang dipakai sebagai uji coba. SWRL rule yang digunakan sebagai uji coba merupakan rule yang telah dijelaskan pada Sub bab Pencarian Relasi *intervalEquals*. Sedangkan *pointer* 3 menunjukkan fakta baru yang berupa data *inference* setelah proses *reasoning*. Hasil tersebut membuktikan bahwa uji coba kevalidan relasi *intervalEquals* telah berhasil.



Gambar 6.16 Pengujian Kevalidan Relasi *intervalEquals*

6.2.2 Pengujian Fungsionalitas

Pengujian fungsionalitas merupakan tahap uji kevalidan fungsional sistem dalam memunculkan fakta-fakta mengenai relasi antar objek-objek warisan budaya yang telah dicari dengan menggunakan ontologi. Pengujian dilakukan dengan melakukan *query* dengan terhadap ontologi dengan menggunakan SPARQL *query*. *Query* dilakukan untuk memunculkan relasi objek-objek warisan budaya berdasarkan masukan yang diberikan sebagai tolok ukur keberhasilan pengujian. Masukan yang diujikan meliputi *before*, *after*, *during*, dan *interval*.

Dalam pengujian ini *query* SPARQL dijalankan dengan menggunakan aplikasi Protégé. Dibutuhkan penelitian lebih lanjut untuk menangani pengujian pada aplikasi iStory berbasis PHP. Hal ini dikarenakan terbatasnya fungsi-fungsi yang disediakan oleh *library* EasyRDF untuk melakukan *query* SPARQL pada *file* lokal.

Pengujian ini membutuhkan *file* RDF hasil *inference* skema ontologi. Studi kasus ini menggunakan 96 *classes*, 287 *object properties*, 24 *datatype properties*, 207 *individuals*. Proses *reasoning* membutuhkan waktu 30 menit sedangkan proses ekspor hasil *reasoning* membutuhkan kurang lebih 11 jam.

6.2.2.1 Pengujian Fungsionalitas *Before*

Pada tahap pengujian fungsionalitas *before* disediakan masukan berupa entitas temporal dari suatu objek warisan budaya. Tujuan dari pengujian ini adalah untuk mengetahui objek-objek yang aktif atau terjadi sebelum entitas temporal yang dipilih sebagai masukan. Rincian pengujian ini dapat dilihat pada Tabel 6.17.

Tabel 6.17 Pengujian Fungsionalitas *Before*

ID	TA-UJ.UC0201
Nama	Pengujian fungsionalitas <i>before</i>

Tujuan Pengujian	Menguji kevalidan <i>query</i> terhadap objek-objek studi kasus relasi <i>before</i> .
Skenario 1	Melihat daftar individu yang seharusnya memiliki relasi <i>before</i> dan tidak.
Kondisi Awal	Hasil <i>reasoning</i> ontologi telah diekspor dan dimuat kembali ke dalam aplikasi Protégé.
Data Uji	Data uji merupakan individual “Abad_11M”.
Langkah Pengujian	<ol style="list-style-type: none"> 1. Pengguna memasukkan <i>query</i> SPARQL melalui aplikasi Protégé. 2. Pengguna menekan tombol <i>Execute</i>.
Hasil Yang Diharapkan	Muncul objek-objek yang aktif atau terjadi sebelum individual yang diujikan.
Hasil Yang Didapat	Objek-objek yang aktif atau terjadi sebelum individual yang diujikan muncul sebagai hasil <i>query</i> .
Hasil Pengujian	Berhasil
Kondisi Akhir	Protégé menampilkan entitas temporal yang terjadi sebelum individual yang diujikan serta objek warisan budaya yang menyertainya.

Hasil pengujian fungsionalitas relasi *before* dapat dilihat pada Gambar 6.17. Ketika uji coba dilakukan objek-objek yang seharusnya aktif atau terjadi sebelum entitas temporal yang dipilih muncul sebagai hasil *query* berdasarkan masukan yang terbaca oleh sistem. *Pointer* 1 menunjukkan *query* SPARQL yang diterapkan dalam pengujian. *Pointer* 2 menunjukkan hasil keluaran dari *query* SPARQL yang diterapkan. Hasil tersebut membuktikan bahwa uji coba fungsionalitas *before* telah berhasil.

<pre> PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> PREFIX owl: <http://www.w3.org/2002/07/owl#> PREFIX xsd: <http://www.w3.org/2001/XMLSchema#> PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#> PREFIX time: <http://www.w3.org/2006/time#> PREFIX cidoc: <http://www.cidoc-crm.org/cidoc-crm#> PREFIX : <http://www.istory.id#> SELECT ?event ?entity WHERE { ?event a time:TemporalEntity; cidoc:P4i_is_time-span_of ?entity; time:before 'Abad_11M' }</pre>	1														
<table> <thead> <tr> <th>event</th><th>entity</th></tr> </thead> <tbody> <tr> <td>'0068098 Pecahan Tembikar / Gerabah'</td><td>'0068098 Pecahan Tembikar / Gerabah'</td></tr> <tr> <td>'Masa Aksara Pallawa'</td><td>'Aksara Pallawa'</td></tr> <tr> <td>'0056737 Pecahan Gelang'</td><td>'0056737 Pecahan Gelang'</td></tr> <tr> <td>'0056788 Pecahan Tembikar/ Gerabah'</td><td>'0056788 Pecahan Tembikar/ Gerabah'</td></tr> <tr> <td>'0056688 Pecahan Tembikar/ Gerabah'</td><td>'0056688 Pecahan Tembikar/ Gerabah'</td></tr> <tr> <td>'0056751 Pecahan Keramik/ Piring'</td><td>'0056751 Pecahan Keramik/ Piring'</td></tr> </tbody> </table>	event	entity	'0068098 Pecahan Tembikar / Gerabah'	'0068098 Pecahan Tembikar / Gerabah'	'Masa Aksara Pallawa'	'Aksara Pallawa'	'0056737 Pecahan Gelang'	'0056737 Pecahan Gelang'	'0056788 Pecahan Tembikar/ Gerabah'	'0056788 Pecahan Tembikar/ Gerabah'	'0056688 Pecahan Tembikar/ Gerabah'	'0056688 Pecahan Tembikar/ Gerabah'	'0056751 Pecahan Keramik/ Piring'	'0056751 Pecahan Keramik/ Piring'	2
event	entity														
'0068098 Pecahan Tembikar / Gerabah'	'0068098 Pecahan Tembikar / Gerabah'														
'Masa Aksara Pallawa'	'Aksara Pallawa'														
'0056737 Pecahan Gelang'	'0056737 Pecahan Gelang'														
'0056788 Pecahan Tembikar/ Gerabah'	'0056788 Pecahan Tembikar/ Gerabah'														
'0056688 Pecahan Tembikar/ Gerabah'	'0056688 Pecahan Tembikar/ Gerabah'														
'0056751 Pecahan Keramik/ Piring'	'0056751 Pecahan Keramik/ Piring'														

Gambar 6.17 Hasil Pengujian Fungsionalitas *Before*

6.2.2.2 Pengujian Fungsionalitas *After*

Pada tahap pengujian fungsionalitas *after* disediakan masukan berupa entitas temporal dari suatu objek warisan budaya. Tujuan dari pengujian ini adalah untuk mengetahui objek-objek yang aktif atau terjadi setelah entitas temporal yang dipilih sebagai masukan. Rincian pengujian ini dapat dilihat pada Tabel 6.18.

Tabel 6.18 Pengujian Fungsionalitas *After*

ID	TA-UJ.UC0202
Nama	Pengujian fungsionalitas <i>after</i>
Tujuan Pengujian	Menguji kevalidan <i>query</i> terhadap objek-objek studi kasus relasi <i>after</i> .
Skenario 1	Melihat daftar individu yang seharusnya memiliki relasi <i>after</i> dan tidak.
Kondisi Awal	Hasil <i>reasoning</i> ontologi telah diekspor dan dimuat kembali ke dalam aplikasi Protégé.
Data Uji	Data uji merupakan individual “Abad_11M”.
Langkah Pengujian	<ol style="list-style-type: none"> 1. Pengguna memasukkan <i>query</i> SPARQL melalui aplikasi Protégé. 2. Pengguna menekan tombol <i>Execute</i>.

Hasil Yang Diharapkan	Muncul objek-objek yang aktif atau terjadi setelah individual yang diujikan.
Hasil Yang Didapat	Objek-objek yang aktif atau terjadi setelah individual yang diujikan muncul sebagai hasil <i>query</i> .
Hasil Pengujian	Berhasil
Kondisi Akhir	Protégé menampilkan entitas temporal yang terjadi setelah individual yang diujikan serta objek warisan budaya yang menyertainya.

Hasil pengujian fungsionalitas relasi *after* dapat dilihat pada Gambar 6.19. Ketika uji coba dilakukan objek-objek yang seharusnya aktif atau terjadi setelah entitas temporal yang dipilih muncul sebagai hasil *query* berdasarkan masukan yang terbaca oleh sistem. *Pointer 1* menunjukkan *query* SPARQL yang diterapkan dalam pengujian. *Pointer 2* menunjukkan hasil keluaran dari *query* SPARQL yang diterapkan. Hasil tersebut membuktikan bahwa uji coba fungsionalitas *after* telah berhasil.



```

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX time: <http://www.w3.org/2006/time#>
PREFIX cidoc: <http://www.cidoc-crm.org/cidoc-crm/>
PREFIX : <http://www.istory.id#>

SELECT ?event ?entity
WHERE
{
    ?event a time:TemporalEntity;
        cidoc:P41_is_time-span_of ?entity;
        time:after :Abad_11M
}

```

Gambar 6.18 Hasil Pengujian Fungsionalitas *After* Bagian

event	entity
'Era Kesultanan Aceh'	'Kesultanan Aceh'
'0056740 Pecahan Keramik/ Mangkuk'	'0056740 Pecahan Keramik/ Mangkuk'
'Perang Diponegoro'	'Perang Diponegoro'
'Era Ken Arok'	'Ken Arok'
'0056741 Pecahan Keramik/ Piring'	'0056741 Pecahan Keramik/ Piring'
'Era Ali Mughayat Syah'	'Ali Mughayat Syah'
'Era Pati Unus'	'Pati Unus'
'Era Raden Patah'	'Raden Patah'
'Era Kesultanan Demak'	'Kesultanan Demak'
'0056736 Pecahan Keramik/ Piring'	'0056736 Pecahan Keramik/ Piring'
'Era Alaidin Muhammad Daud Syah'	'Alaidin Muhammad Daud Syah'
'Masa Aksara Sunda Kuna'	'Aksara Sunda Kuna'
'Era Kesultanan Samudera Pasai'	'Kesultanan Samudera Pasai'

Gambar 6.19 Hasil Pengujian Fungsionalitas *After* Bagian 2

6.2.2.3 Pengujian Fungsionalitas *During*

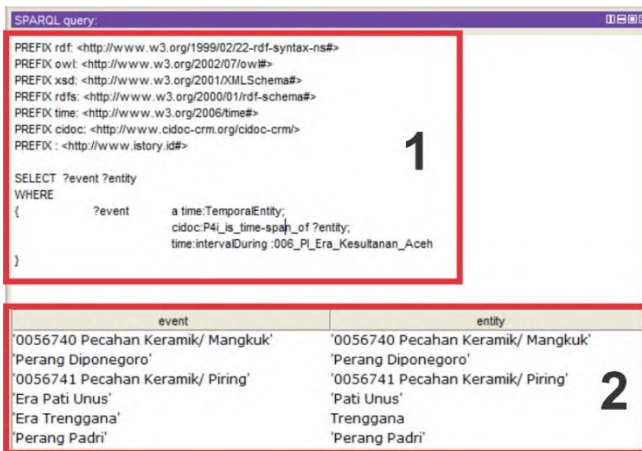
Pada tahap pengujian fungsionalitas *during* disediakan masukan berupa entitas temporal dari suatu objek warisan budaya. Tujuan dari pengujian ini adalah untuk mengetahui objek-objek yang aktif atau terjadi dalam kurun waktu tertentu yang dipilih sebagai masukan. Rincian pengujian ini dapat dilihat pada Tabel 6.19.

Tabel 6.19 Pengujian Fungsionalitas *During*

ID	TA-UJ.UC0203
Nama	Pengujian fungsionalitas <i>during</i>
Tujuan Pengujian	Menguji kevalidan <i>query</i> terhadap objek-objek studi kasus relasi <i>during</i> .
Skenario 1	Melihat daftar individu yang seharusnya aktif atau terjadi dalam kurun waktu tertentu dan tidak.
Kondisi Awal	Hasil <i>reasoning</i> ontologi telah diekspor dan dimuat kembali ke dalam aplikasi Protégé.
Data Uji	Data uji merupakan individual "006_PI_Era_Kesultanan_Aceh".
Langkah Pengujian	<ol style="list-style-type: none"> 1. Pengguna memasukkan <i>query</i> SPARQL melalui aplikasi Protégé. 2. Pengguna menekan tombol <i>Execute</i>.

Hasil Yang Diharapkan	Muncul objek-objek yang aktif atau terjadi dalam kurun waktu individual yang diujikan.
Hasil Yang Didapat	Objek-objek yang aktif atau terjadi dalam kurun waktu individual yang diujikan muncul sebagai hasil <i>query</i> .
Hasil Pengujian	Berhasil
Kondisi Akhir	Protégé menampilkan entitas temporal yang terjadi dalam kurun waktu individual yang diujikan serta objek warisan budaya yang menyertainya.

Hasil pengujian fungsionalitas relasi *during* dapat dilihat pada Gambar 6.20. Ketika uji coba dilakukan objek-objek yang seharusnya aktif atau terjadi dalam kurun waktu entitas temporal yang dipilih muncul sebagai hasil *query* berdasarkan masukan yang terbaca oleh sistem. *Pointer 1* menunjukkan *query* SPARQL yang diterapkan dalam pengujian. *Pointer 2* menunjukkan hasil keluaran dari *query* SPARQL yang diterapkan. Hasil tersebut membuktikan bahwa uji coba fungsionalitas *during* telah berhasil.



SPARQL query.

```

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX time: <http://www.w3.org/2006/time#>
PREFIX cidoc: <http://www.cidoc-crm.org/cidoc-crm#>
PREFIX : <http://www.istory.id#>

SELECT ?event ?entity
WHERE
{
    ?event      a time:TemporalEntity;
               cidoc:P4i_is_time-span_of ?entity;
               time:intervalDuring :006_Pl_Era_Kesultanan_Aceh
}

```

event	entity
'0056740 Pecahan Keramik/ Mangkuk'	'0056740 Pecahan Keramik/ Mangkuk'
'Perang Diponegoro'	'Perang Diponegoro'
'0056741 Pecahan Keramik/ Piring'	'0056741 Pecahan Keramik/ Piring'
'Era Pati Unus'	'Pati Unus'
'Era Trenggana'	Trenggana
'Perang Padri'	'Perang Padri'

Gambar 6.20 Hasil Pengujian Fungsionalitas *During*

6.2.2.4 Pengujian Fungsionalitas *Interval*

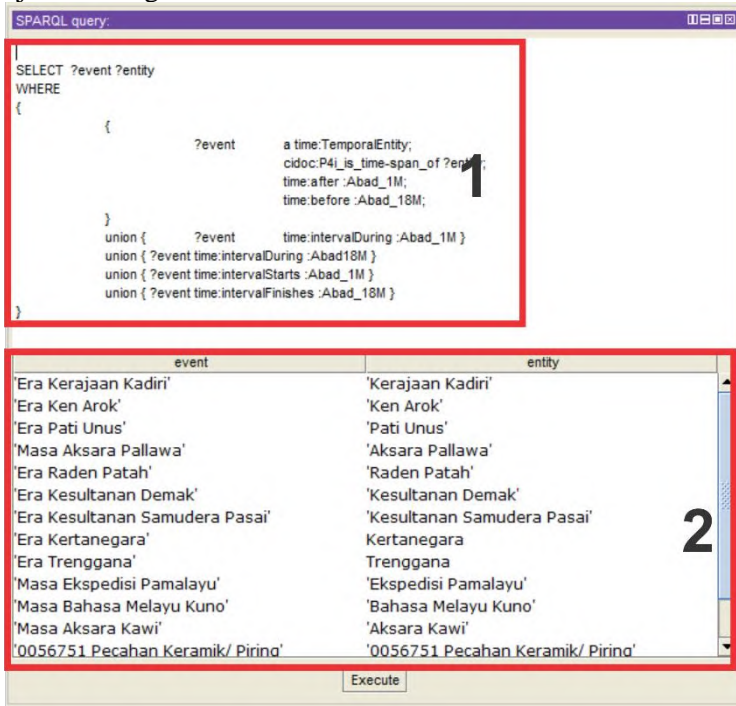
Pada tahap pengujian fungsionalitas *interval* disediakan masukan berupa dua entitas temporal dari objek warisan budaya. Tujuan dari pengujian ini adalah untuk mengetahui objek-objek yang aktif atau terjadi dalam rentang waktu antara dua entitas temporal yang dipilih sebagai masukan. Rincian pengujian ini dapat dilihat pada Tabel 6.20.

Tabel 6.20 Pengujian Fungsionalitas *Interval*

ID	TA-UJ.UC0204
Nama	Pengujian fungsionalitas <i>interval</i>
Tujuan Pengujian	Menguji kevalidan <i>query</i> terhadap objek-objek studi kasus relasi <i>interval</i> .
Skenario 1	Melihat daftar individu yang seharusnya aktif atau terjadi dalam rentang waktu antara dua entitas temporal dan tidak.
Kondisi Awal	Hasil <i>reasoning</i> ontologi telah diekspor dan dimuat kembali ke dalam aplikasi Protégé.
Data Uji	Data uji merupakan individual “Abad_1M” sebagai titik awal dan “Abad_18M” sebagai titik akhir.
Langkah Pengujian	<ol style="list-style-type: none"> 1. Pengguna memasukkan <i>query</i> SPARQL melalui aplikasi Protégé. 2. Pengguna menekan tombol <i>Execute</i>.
Hasil Yang Diharapkan	Muncul objek-objek yang aktif atau terjadi dalam rentang waktu antara dua individual yang diujikan.
Hasil Yang Didapat	Objek-objek yang aktif atau terjadi dalam rentang waktu antara dua individual yang diujikan muncul sebagai hasil <i>query</i> .
Hasil Pengujian	Berhasil
Kondisi Akhir	Protégé menampilkan entitas temporal yang terjadi dalam rentang waktu dua individual yang diujikan serta objek warisan budaya yang menyertainya.

Hasil pengujian fungsionalitas relasi *interval* dapat dilihat pada Gambar 6.21. Ketika uji coba dilakukan objek-objek yang seharusnya aktif atau terjadi rentang waktu antara dua entitas temporal yang dipilih muncul sebagai hasil *query*

berdasarkan masukan yang terbaca oleh sistem. *Pointer 1* menunjukkan *query* SPARQL yang diterapkan dalam pengujian. *Pointer 2* menunjukkan hasil keluaran dari *query* SPARQL yang diterapkan. Hasil tersebut membuktikan bahwa uji coba fungsionalitas *interval* telah berhasil.



The screenshot shows a SPARQL query interface. The query is as follows:

```

SELECT ?event ?entity
WHERE
{
    {
        ?event a time:TemporalEntity;
        cidoc:P4i_is_time-span_of ?entity;
        time:after :Abad_1M;
        time:before :Abad_18M;
    }
    union {
        ?event time:intervalDuring :Abad_1M }
    union { ?event time:intervalDuring :Abad_18M }
    union { ?event time:intervalStarts :Abad_1M }
    union { ?event time:intervalFinishes :Abad_18M }
}

```

The results are displayed in a table with two columns: **event** and **entity**. The results are as follows:

event	entity
'Era Kerajaan Kadiri'	'Kerajaan Kadiri'
'Era Ken Arok'	'Ken Arok'
'Era Pati Unus'	'Pati Unus'
'Masa Aksara Pallawa'	'Aksara Pallawa'
'Era Raden Patah'	'Raden Patah'
'Era Kesultanan Demak'	'Kesultanan Demak'
'Era Kesultanan Samudera Pasai'	'Kesultanan Samudera Pasai'
'Era Kertanegara'	Kertanegara
'Era Trenggana'	Trenggana
'Masa Ekspedisi Pamalayu'	'Ekspedisi Pamalayu'
'Masa Bahasa Melayu Kuno'	'Bahasa Melayu Kuno'
'Masa Aksara Kawi'	'Aksara Kawi'
'0056751 Pecahan Keramik/ Pirino'	'0056751 Pecahan Keramik/ Pirino'

Gambar 6.21 Hasil Pengujian Fungsionalitas *Interval*

6.3 Evaluasi Pengujian

Pada subbab ini akan diberikan hasil evaluasi dari pengujian-pengujian yang telah dilakukan. Evaluasi yang diberikan adalah evaluasi pengujian skema ontologi dan pengujian fungsional. Rangkuman mengenai hasil pengujian skema ontologi dapat dilihat pada Tabel 6.21 dan hasil pengujian fungsionalitas dapat dilihat pada Tabel 6.22.

Berdasarkan data pada tabel tersebut, semua skenario pengujian berhasil. Sehingga bisa ditarik disimpulkan bahwa model yang dibuat telah sesuai dengan yang diharapkan. Akan tetapi, proses komputasional apabila terjadi penambahan data akan memakan waktu yang relatif lama karena harus melewati tahapan *reasoning* dan ekspor *inference* data.

Tabel 6.21 Evaluasi Pengujian Ontologi

ID	Nama	Skenario	Hasil
TA-UJ.UC0101	Pengujian kevalidan relasi <i>before</i>	Skenario 1	Berhasil
TA-UJ.UC0102	Pengujian kevalidan relasi <i>after</i>	Skenario 1	Berhasil
TA-UJ.UC0103	Pengujian kevalidan relasi <i>equals</i>	Skenario 1	Berhasil
TA-UJ.UC0104	Pengujian kevalidan relasi <i>intervalMeets</i>	Skenario 1	Berhasil
TA-UJ.UC0105	Pengujian kevalidan relasi <i>intervalMetBy</i>	Skenario 1	Berhasil
TA-UJ.UC0106	Pengujian kevalidan relasi <i>intervalBefore</i>	Skenario 1	Berhasil
TA-UJ.UC0107	Pengujian kevalidan relasi <i>intervalAfter</i>	Skenario 1	Berhasil
TA-UJ.UC0108	Pengujian kevalidan relasi <i>intervalDuring</i>	Skenario 1	Berhasil
TA-UJ.UC0109	Pengujian kevalidan relasi <i>intervalContains</i>	Skenario 1	Berhasil
TA-UJ.UC0110	Pengujian kevalidan relasi <i>intervalStarts</i>	Skenario 1	Berhasil
TA-UJ.UC0111	Pengujian kevalidan relasi <i>intervalStartedBy</i>	Skenario 1	Berhasil
TA-UJ.UC0112	Pengujian kevalidan relasi <i>intervalFinishes</i>	Skenario 1	Berhasil
TA-UJ.UC0113	Pengujian kevalidan relasi <i>intervalFinishedBy</i>	Skenario 1	Berhasil
TA-UJ.UC0114	Pengujian kevalidan relasi <i>intervalOverlaps</i>	Skenario 1	Berhasil

TA-UJ.UC0115	Pengujian kevalidan relasi <i>intervalOverlappedBy</i>	Skenario 1	Berhasil
TA-UJ.UC0116	Pengujian kevalidan relasi <i>intervalEquals</i>	Skenario 1	Berhasil

Tabel 6.22 Evaluasi Pengujian Fungsionalitas

ID	Nama	Skenario	Hasil
TA-UJ.UC0201	Pengujian fungsionalitas <i>before</i>	Skenario 1	Berhasil
TA-UJ.UC0202	Pengujian fungsionalitas <i>after</i>	Skenario 1	Berhasil
TA-UJ.UC0203	Pengujian fungsionalitas <i>during</i>	Skenario 1	Berhasil
TA-UJ.UC0204	Pengujian fungsionalitas <i>interval</i>	Skenario 1	Berhasil

BAB VII

KESIMPULAN DAN SARAN

Pada bab ini dijelaskan mengenai kesimpulan dari hasil uji coba yang telah dilakukan dan saran mengenai hal-hal yang masih bisa untuk dikembangkan dari tugas akhir ini.

7.1 Kesimpulan

Dari hasil pengamatan selama proses perancangan, implementasi dan pengujian perangkat lunak yang dilakukan, dapat diambil kesimpulan sebagai berikut:

1. Studi kasus pencarian relasi waktu antar objek warisan budaya Indonesia mampu dimodelkan dengan penggabungan serta pengembangan skema ontologi OWL-Time dan CIDOC-CRM.
2. Relasi antar individu dapat dicari dengan menggunakan SWRL *rule* dengan menyimpan informasi waktu dalam format *dateTime*.
3. Hasil pencarian relasi waktu antar objek warisan budaya dapat ditampilkan ke dalam halaman web dengan memanfaatkan *library* EasyRDF.

7.2 Saran

Berikut merupakan beberapa saran untuk pengembangan sistem di masa yang akan datang. Saran-saran ini didasarkan pada hasil perancangan, implementasi dan pengujian yang telah dilakukan.

1. Perancangan *query* SPARQL untuk membantu ekstraksi data dari DBpedia.
2. Perancangan skema penyimpanan dengan format lain seperti *gYear*, *gMonth*, dan *gDate*.

[Halaman ini sengaja dikosongkan]

DAFTAR PUSTAKA

- [1] A. D. Karmadi, "Budaya Lokal Sebagai Warisan Budaya dan Upaya Pelestariannya," *Dialog Budaya Daerah Balai Pelestarian Sejarah dan Nilai Tradisional Yogyakarta*, 2007.
- [2] L. Petnga and M. Austin, "Ontologies of Time and Time-based Reasoning for MBSE of Cyber-Physical Systems," *Procedia Computer Science*, vol. 16, pp. 403-412, 2013.
- [3] R. Neches, "Enabling technology for knowledge sharing," *AI magazine* 12.3, p. 36, 1991.
- [4] T. R. Gruber and R. G. Olsen, "An ontology for engineering mathematics," *Fourth International Conference on Principles of Knowledge Representation and Reasoning*, 1994.
- [5] L. Stojanovic, S. Staab and R. Studer, "eLearning based on the Semantic Web," *WebNet2001-World Conference on the WWW and Internet*, 2001.
- [6] N. Ibrahim, "Pengembangan aplikasi Semantic Web untuk membangun web yang lebih cerdas," *Jurnal Informatika*, vol. 3.1, p. 27, 2011.
- [7] J. R. Hobbs and F. Pan, "Time Ontology in OWL," *W3C working draft*, vol. 27, p. 133, 2006.
- [8] N. Crofts, "Definition of the CIDOC conceptual reference model," *ICOM/CIDOC Documentation Standards Group. CIDOC CRM Special Interest Group* 5, 2011.
- [9] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosz and M. Dean, "SWRL: A Semantic Web Rule Language Combining OWL and RuleML," *W3C Member submission*, vol. 21, p. 79, 2004.
- [10] E. Anagnostopoulos, S. Sotiris Batsakis and E. Petrakis, "CHRONOS: A Reasoning Engine for Qualitative

Temporal Information in OWL," *Procedia Computer Science*, vol. 22, pp. 70-77, 2013.

- [11] M. Wolf and C. Wicksteed, "Date and Time Formats," *W3C NOTE NOTE-datetime-19980827*, 1997.

BIODATA PENULIS



Alief Yoga Priyanto atau biasa dipanggil Alief dilahirkan di Solo pada tanggal 7 Mei 1994 dan dibesarkan di Solo. Penulis adalah anak pertama dari tiga bersaudara.

Penulis menempuh pendidikan di SD Muhammadiyah 1 (1999-2006), SMP N 4 Surakarta(2006-2009), dan SMA N 3 Surakarta (2009-2012). Setelah lulus SMA penulis melanjutkan ke jenjang perkuliahan di Jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember Surabaya. Bidang Studi yang diambil oleh penulis pada saat kuliah di Teknik Informatika ITS adalah Manajemen Informasi.

Penulis dapat dihubungi melalui alamat *email* aliefyp@gmail.com.